Contents

1 Data Processing Pipeline 3
  1.1 Processing stages for data projects ......................................................... 3
  1.2 An introduction to the data pipeline .......................................................... 3
The School of Data Handbook is a companion text to the School of Data. Its function is something like a traditional textbook - it will provide the detail and background theory to support the School of Data courses and challenges.

The Handbook should be accessible to all learners. It comes with a Glossary explaining the important terms and concepts. If you stumble across an unexplained term or a concept that requires more explanation, please do get in touch.
Data Processing Pipeline

The Handbook will guide you through the key stages of a data project. These stages can be thought of as a pipeline, or a process.

1.1 Processing stages for data projects

While there are many different types of data, almost all processing can be expressed as a set of incremental stages. The most common stages include data acquisition, extraction, cleaning, transformation, integration, analysis and presentation. Of course, with many smaller projects, not each of these stages may be necessary.

1.2 An introduction to the data pipeline

- **Acquisition** describes gaining access to data, either through any of the methods mentioned above or by generating fresh data, e.g. through a survey or observations.

- In the **extraction** stage, data is converted from whatever input format has been acquired (e.g. XLS files, PDFs or even plain text documents) into a form that can be used for further processing and analysis. This often involves loading data into a database system, such as MySQL or PostgreSQL.

- **Cleaning and transforming** the data often involves removing invalid records and translating all the columns to use a sane set of values. You may also combine two different datasets into a single table, remove duplicate entries or apply any number of other normalizations. As you acquire data, you will notice that such data often has many inconsistencies: names are used inconsistently, amounts will be stated in badly formatted numbers, while some data may not be usable at all due to file corruptions. In short: data always needs to be cleaned and processed. In fact, processing, augmenting and cleaning the data is very likely to be the most time- and labour-intensive aspect of your project.

- **Analysis of data** to answer particular questions we will not describe in detail in the following chapters of this book. We presume that you are already the experts in working with your data and using e.g. economic models to answer your questions. The aspects of analysis which we do hope to cover here are automated and large-scale analysis, showing tips and tricks for getting and using data, and having a machine do a lot of the work, for example: network analysis or natural language processing.

- **Presentation of data** only has impact when it is packaged in an appropriate way for the audiences it needs to aim at.
As you model a data pipeline, it is important to take care that each step is well documented, granular and - if at all possible - automated.

We will also cover best practice guidelines for data projects - which may not fit into any particular categories, but are very important parts of any data work!

1.2.1 Table of Contents

Courses

Welcome to the School of Data courses. School of Data produces two types of material:

Data might sound scary sometimes - don't worry, we're there to guide you through! If you have any questions about the material, simply drop us a line via the mailing list

Data Fundamentals

The Data Fundamental modules provide a solid overview over the workflow with data guiding you from what data is, to how to make your data tell a story. The courses listed below should be seen as a whole, a quick overview of the elements involved in working with data.

What is Data?

Introduction Welcome to the beginners course of the School of Data. In this course we will cover the basics of data wrangling and visualization and will discover and tell a story in a dataset.

In this module, you will learn where to start looking for data. We begin with an introduction to some of the basics of data – key terms like qualitative, quantitative, machine-readable, discrete and continuous data, which crop up again and again for Data Wranglers.

Most things start with a question Most people don’t just wrangle data for fun. They have a story to tell or a problem to solve.

Often you will start with a question in mind. This could be anything from: ‘How often does the sun shine in my hometown?’ to ‘How does my government spend its money? And where do they get it from?’. A question is a good starting point for exploring your data - it makes you focused and helps you to detect interesting patterns in the data. Understanding for whom your question is interesting will also help you to define the audience you need to work for, and will help you to shape your story.

What if you start without a question? You’re just exploring. If you find something that looks interesting in your dataset, you can start examining it as if this was the question you had in mind. Sometimes patterns in data can be explained by investigating what causes the patterns. This is often a story worth telling.

Whether you began with a question or not, you should always keep your eyes open for unexpected patterns, unusual results, or anything that surprises you. Often, the most interesting stories aren’t the ones you were looking for.

In this course we will start with a question and then explore a dataset with this question in mind. We will also roam around and explore whether there is something interesting hidden in the data.

The question we will focus on for the Data Fundamentals Course will be: How does healthcare spending influence life expectancy?

Task: Think of a question you would like to answer using data.
What is Data?  Data is all around us. But what exactly is it? Data is a value assigned to a thing. Take for example the balls in the picture below.

Figure 1.1: Golf balls at a market (CC) by Kaptain Kobold on Flickr.

What can we say about these? They are golf balls, right? So one of the first data points we have is that they are used for golf. Golf is a category of sport, so this helps us to put the ball in a taxonomy. But there is more to them. We have the colour: “white”, the condition “used”. They all have a size, there is a certain number of them and they probably have some monetary value, and so on.

Even unremarkable objects have a lot of data attached to them. You too: you have a name (most people have given and family names) a date of birth, weight, height, nationality etc. All these things are data.

In the example above, we can already see that there are different types of data. The two major categories are qualitative and quantitative data.

**Qualitative data** is everything that refers to the quality of something: A description of colours, texture and feel of an object, a description of experiences, and interview are all qualitative data.

**Quantitative data** is data that refers to a number. E.g. the number of golf balls, the size, the price, a score on a test etc.

However there are also other categories that you will most likely encounter:

**Categorical data** puts the item you are describing into a category: In our example the condition “used” would be categorical (with categories such as “new”, “used”, “broken” etc.)

**Discrete data** is numerical data that has gaps in it: e.g. the count of golf balls. There can only be whole numbers of golf ball (there is no such thing as 0.3 golf balls). Other examples are scores in tests (where you receive e.g. 7/10) or shoe sizes.

**Continuous data** is numerical data with a continuous range: e.g. size of the golfballs can be any value (e.g. 10.53mm or 10.54mm but also 10.536mm), or the size of your foot (as opposed to your shoe size, which is discrete): In continuous data, all values are possible with no gaps in between.

**Task:** Take the example of golf balls: can you find data of all different categories?

From Data to Information to Knowledge.  Data, when collected and structured suddenly becomes a lot more useful. Let’s do this in the table below.

<table>
<thead>
<tr>
<th>Colour</th>
<th>White</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Sport - Golf</td>
</tr>
<tr>
<td>Condition</td>
<td>Used</td>
</tr>
<tr>
<td>Diameter</td>
<td>43mm</td>
</tr>
<tr>
<td>Price (per ball)</td>
<td>$0.5 (AUD)</td>
</tr>
</tbody>
</table>

But each of the data values is still rather meaningless by itself. To create information out of data, we need to interpret that data.

Let’s take the size: A diameter of 43mm doesn’t tell us much. It is only meaningful when we compare it to other things. In sports there are often size regulations for equipment. The minimum size for a competition golf ball is 42.67mm. Good, we can use that golf ball in a competition. This is information. But it still is not knowledge. Knowledge is created when the information is learned, applied and understood.

Unstructured vs. Structured data
Data Wrangling Handbook, Release 0.1

Data for Humans  A plain sentence - “we have 5 white used golf balls with a diameter of 43mm at 50 cents each” - might be easy to understand for a human, but for a computer this is hard to understand. The above sentence is what we call unstructured data. Unstructured has no fixed underlying structure - the sentence could easily be changed and it’s not clear which word refers to what exactly. Likewise, PDFs and scanned images may contain information which is pleasing to the human-eye as it is laid-out nicely, but they are not machine-readable.

Data for Computers  Computers are inherently different from humans. It can be exceptionally hard to make computers extract information from certain sources. Some tasks that humans find easy are still difficult to automate with computers. For example, interpreting text that is presented as an image is still a challenge for a computer. If you want your computer to process and analyse your data, it has to be able to read and process the data. This means it needs to be structured and in a machine-readable form.

One of the most commonly used formats for exchanging data is CSV. CSV stands for comma separated values. The same thing expressed as CSV can look something like:

```
“quantity”, “color”, “condition”, “item”, “category”, “diameter (mm)”, “price per unit (AUD)”
5,”white”,”used”,”ball”,”golf”,43,0.5
```

This is way simpler for your computer to understand and can be read directly by spreadsheet software. Note that words have quotes around them: This distinguishes them as text (string values in computer speak) - whereas numbers do not have quotes. It is worth mentioning that there are many more formats out there that are structured and machine readable.

Task: Think of the last book you read. What data is connected to it and how would you make it structured data?

Summary  In this tutorial we explored some of the essential concepts that crop up again and again in discussions of data. What discussed what data is, and how it is structured. In the next Tutorial we will look at data sources and how to get hold of data.

Extra Reading

1. When you get a new dataset, should you dive in / should you have a hypothesis ready to go? Caelainn Barr, an award winning journalist explains how she approaches new data sources. [http://datajournalismhandbook.org/1.0/en/understanding_data_4.html](http://datajournalismhandbook.org/1.0/en/understanding_data_4.html)

2. Overview of common file formats in the open data handbook.

Quiz  Take the following quiz to check whether you understood basic data categories.

Data provenance  Good documentation on data provenance (the origin and history of a dataset) can be compared to the chain of custody which is maintained for criminal investigations: each previous owner of a dataset must be identified, and they are held accountable for the processing and cleaning operations they have performed on the data. For Excel spreadsheets this would include writing down the steps taken in transforming the data, while advanced data tools (such as Open Refine, formerly Google Refine), often provide methods of exporting machine-readable data containing processing history. Any programs that have been written to process the data should be available when users access your end result and shared as open-source code on a public code sharing site such as GitHub.

Tools for documenting your data work  Documenting the transformations you perform on your data can be as simple as a detailed prose explanation and a series of spreadsheets that represent key, intermediate steps. But there are also a few products out there that are specifically geared towards helping you do this. Socrata is one platform that helps you perform transforms on spreadsheet-like data and share them with others easily. You can also use the Data Hub (pictured below), an open source platform that allows for several versions of a spreadsheet to be collected together into one dataset, and also auto-generates an API to boot.
If you want to get really smart, you may like to try using version control for your data work. Using version control will track every change made to the data, and allow you to easily roll back if there are any mistakes. Software Carpentry offer a good introduction to the topic.

**Tips for documenting your work**

1. Link to the original data and mention where you got it from. This is important to show exactly where the original source was.

2. Show and describe everything that you change in the data along the way. This is equally important, both for you and for others to be able to check your work. (See how in the screenshot above - we have adjusted units for currency and explained what we have done at every step).

**Finding Data**

**Introduction**  
Now we know what data is and the questions we’re interested in, we’re ready to go out and hunt for it online.

In this tutorial, you will learn where to start looking for data. In this course, we will then look at different ways of getting hold of data, before setting you loose to find data yourselves!

**Data Sources**  
There are three basic ways of getting hold of data:

1. **Finding data** - this involves searching and finding data that has already been released

2. **Getting hold of more data** - asking for ‘new’ data from official sources e.g. through Freedom of Information requests. Sometimes data is public on a website but there is not a download link to get hold of it in bulk - but don’t give up! This data can be libered with what datawranglers call scraping.

3. **Collecting data yourself** - This means gathering data and entering it into a database or a spreadsheet - whether you work alone or collaboratively.

In this tutorial we’ll focus on finding data that already has been released. We will deal with getting more data and collecting data yourself in future courses.

**Step 1: Identify your Data Source**  
Many sources frequently release data for public use. Some examples:

1. **Government** In recent years governments have begun to release some of their data to the public. Many governments host special (open) government data platforms for the data they create. For example the UK government started data.gov.uk to release their datasets. Similar data portals exist in the US, Brazil and Kenya - and in many other countries! Does your country have an open data portal (Datacatalogs.org is a good starting point)?

2. **Organisations** Other sources of data are large organisations. The World Bank and the World Health Organization for example regularly release reports and data sets.

3. **Science** Scientific projects and institutions release data to the scientific community and the general public. Open data is produced by NASA for example, and many specific disciplines have their own data repositories, some of which are open. More and more initiatives exist trying to provide access to already published data (e.g. Dryad)

To help people to find data, projects like the Open Access Directory’s data repository list or the Open Knowledge Foundation’s datahub.io have been started. They aim either to collect data sources, or collect together different data sets from various sources. Also the School of Data is building a list of data sources with your help!

**Task:** Are there any data sources we missed? What kind of data would be most interesting to you? Add them to our list of data sources.
Step 2: Getting data in the format you need it  In the “What is Data” course, we talked briefly about the importance of machine-readable data. You’ll save yourself a lot of trouble and time in working with the data if you get hold of data in the correct format initially. Here’s a handy tip for how to tell Google which format you are looking for.

Using data to answer your question  Now that you have an overview of some of the key concepts related to data, it’s time to start hunting for your own! Over the next courses in the Data Fundamentals series, we will be further exploring the question we posed ourselves in the What is Data Course? How does healthcare spending influence life expectancy? To get the data for this course, please see our recipe on Getting Data from the World Bank.

Task: If you found your own alternative data to answer this question, congratulations! Take a moment to upload it to the DataHub - and have a browse to see what other School of Data learners have found.

Extension Task: Explore the web, and see what open data you can find. If you find something really interesting and think of an exciting question it could help to address, tweet it to @SchoolofData - or write a short post for the School of Data blog.

Summary  In this tutorial we discussed how we get the data to answer our question. We explored different ways of accessing data sources and introduced several resources listing different data portals and search engines.

At the beginning of Data Fundamentals, we posed ourselves a question: ‘How does healthcare spending influence life expectancy?’, and by following the recipe, have found a dataset from the World Bank that will help us to answer that question.

Extra Reading
1. How to get data from the world bank data portal
2. How to upload data to datahub.io http://vimeo.com/45913395
3. The Data Journalism Handbook has lots of handy tips for finding useful data sources in a “Five Minute Field Guide”

Quiz  Take the following quiz to check whether you understood where to find data.

Sort and Filter: The basics of spreadsheets

Introduction  The most basic tool used for data wrangling is a spreadsheet. Data contained in a spreadsheet is in a structured, machine-readable format and hence can quickly be sorted and filtered. In other recipes in the handbook, you’ll learn how to use the humble spreadsheet as a power tool for carrying out simple sums (finding the total, the average etc.), applying bulk processes, or pulling out different graphs and charts.

By the end of the module, you will have learned how to download data, how to import it into a spreadsheet, and how to begin cleaning and interpreting it using the ‘sort’ and ‘filter’ functions.

Spreadsheets: An Overview  Nowadays spreadsheets are widespread so a lot of people are familiar with them already. A variety of spreadsheet programs and applications exist. For example Microsoft’s Office package comes with Excel, the OpenOffice package comes with Calc and so on. Not surprisingly, Google decided to add spreadsheets to their documents package. Since it does not require you to purchase or install any additional software, we will be using Google Spreadsheets for this course.

Depending on what you want to do you might consider using different spreadsheet software. Here are some of the considerations you might make when picking your weapon of choice:
Creating a spreadsheet and uploading data  In this course we will use Google docs for our data-wrangling - it allows you to start right away without need of installing software. Since the data we are working with is already public we also don’t need to worry about the fact that it is not stored on our local drive.

Walktrough: Creating a Spreadsheet and uploading data.

1. Head over to Google docs.
2. If you are not yet logged in to Google docs, you need to login.
3. The first step is going to be creating a new spreadsheet.
4. Do this by clicking the create button to the left and select spreadsheet.
5. Doing so will create a new spreadsheet for you.
6. Let’s upload some data.
7. You will need the file we downloaded from the World Bank in the last tutorial. If you haven’t done the tutorial or lost the file: download it here .
8. In your spreadsheet select import from the file menu. This will open a dialog for you.
9. Select the file you downloaded.
10. Don’t forget to select insert new sheets, and click import

Navigating and using the Spreadsheet  Now we loaded some data let’s deal with the basics of spreadsheets. A spreadsheet is basically a table of “cells” in which you can input data. The cells are organized in “rows” and “columns”. Typically rows are labeled by numbers, columns by letters. This also means cells can be addressed by their “column” and “row” coordinates. The cell A1 denotes the cell in the first row in the first column, A2 the one in the second row, B1 the one in the second column and so on.

To enter or change data in a cell click on it and start typing - this will change the contents of the cell. Basic navigation can be done this way or via keyboard. Find a list of keyboard shortcuts good to know below:

<table>
<thead>
<tr>
<th>Spreadsheet</th>
<th>Google Spreadsheets</th>
<th>Open(Libre)Office</th>
<th>Microsoft Excel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>Free (as in Beer)</td>
<td>Free (as in Freedom)</td>
<td>Commercial</td>
</tr>
<tr>
<td>Data Storage</td>
<td>Google Drive</td>
<td>Your hard disk</td>
<td>Your hard disk</td>
</tr>
<tr>
<td>Needs Internet</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Installation required</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Collaboration</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Sharing results</td>
<td>Easy</td>
<td>Harder</td>
<td>Harder</td>
</tr>
<tr>
<td>Visualizations</td>
<td>Large range</td>
<td>Basic charts</td>
<td>Basic charts</td>
</tr>
<tr>
<td>Key or Combination</td>
<td>What it does</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tab</td>
<td>End input on the current cell and jump to the cell right to the current one</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enter</td>
<td>End input and jump to the next row (This will try to be intelligent, so if you’re entering multiple columns, it will jump to the first column you are entering</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Up</td>
<td>Move to the cell one row up</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Down</td>
<td>Move to the cell one row down</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>Move to the cell left</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Right</td>
<td>Move to the cell on the Right</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ctrl+&lt;direction&gt;</td>
<td>Move to the outermost cell in the direction given</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shift+&lt;direction&gt;</td>
<td>Select the current cell and the cell in &lt;direction&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ctrl+Shift+&lt;direction&gt;</td>
<td>Select all cells from the current to the outermost cell in &lt;direction&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ctrl+c</td>
<td>Copy – copies the selected cells into the clipboard</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ctrl+v</td>
<td>Paste – pastes the clipboard</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ctrl+x</td>
<td>Cut – cuts the selected cells into the clipboard and removes them from their original position</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ctrl+z</td>
<td>Undo – undoes the last change you made</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ctrl+y</td>
<td>Redo – undoes an undo</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Tip:** Practice a bit, and you will find that you will become a lot faster using the keyboard than the mouse!

**Locking Rows and Columns** The spreadsheet we are working on is quite large. You will notice, that while scrolling the column with the column labels will frequently disappear, leaving you quite lost. The same with the country names. To avoid this you can “lock” rows and columns so they don’t disappear.

**Walkthrough: Locking the top row**

1. Go to the Spreadsheet with our data and scroll to the top.
2. On the top left, where the column and row labels are you’ll see a small striped area.
3. Hover over the striped bar on top of box showing row “1”. A hand shaped cursor should appear, click and drag it down one row.
4. Your result should look like this:
5. Try scrolling – notice how the top row remains fixed?

**Sorting Data** The first thing to do when looking at a new dataset is to orient yourself. This involves at looking at maximum/minimum values and sorting the data so it makes sense. Let’s look at the columns. We have data about the GDP, healthcare expenditure and life expectancy. Now let’s explore the range of data by simply sorting.

**Walkthrough: Sorting a dataset**

1. Select the whole sheet you want to sort. Do this by clicking on the right upper grey field, between the row and column names.
2. Select “Sort Range...” from the “Data” menu – this will open an additional Selection
3. Check the “Data has header row” checkbox
4. Select the column you want to sort by in the dropdown menu
5. Try to sort by GDP – Which country has the lowest?
6. Try again with different values, can you sort ascending and descending?
Tip: Be careful! A common mistake is to forget to select all the data. If you sort without selecting all the data, the rows will no longer match up.

A version of this recipe can also be found in the Handbook.

Filtering Data  The next thing commonly done with datasets is to filter out the values you don’t want to see. Did you notice that some “Country Names” are actually not countries? You’ll find things like “World”, “North America” and “Arab World”. Let’s filter them out.

Walkthrough: Filtering Data

1. Select the whole table.
2. Select “Filter” from the “Data” menu.
3. You now should see triangles next to the column names in the first row.
4. Click on the triangle next to country name.
5. you should see a long list of country names in the box.
6. Find those that are not a country and click on them (the green check mark will disappear).
7. Now you have successfully filtered your dataset.
8. Go ahead and play with it - the data will not be deleted, it’s just not displayed.

A version of this recipe can also be found in the Handbook.

Summary  In this module we talked about basic spreadsheet skills. We talked about data entry and how to sort and filter data using a spreadsheet program. In the next course we will talk about data analysis and introduce you to formulas.

Further Reading and References

1. Google help on spreadsheets

Quiz  Check your sorting and filtering skills with the following quiz.

‘But what does it mean?’: Analyzing data (spreadsheets continued)

Introduction  Once you have cleaned and filtered your dataset - it’s time for analysis. Analysing data helps us to learn what our data might mean and helps us to extract answers to our questions from the dataset.

Look at the data we imported. (In case you didn’t finish the previous tutorial, don’t worry. You can copy a sample spreadsheet here).

This is World Bank data containing GDP, population, health expenditure and life expectancy for the years 2000-2011. Take a moment to have a look at the data. It’s pretty interesting - what could it tell us?

Task: Brainstorm ideas. What could you investigate using this data?

Here are some ideas we came up with:

1. How much (in USD) is spent on healthcare in total in each country?
2. How much (in USD) is spent per capita in each country?
3. In which country is the most spent per person? In which country is the least spent? What is the average for each continent? For the world?

4. What is the relationship between public and private health expenditure in each country? Where do citizens spend more (private expenditure)? Where does the state spend more (public expenditure)?

5. Is there a relationship between expenditure on healthcare and average life expectancy?

6. Does it make any difference if the expenditure is public or private?

**NOTE:** With these last two questions, you have to be really careful. Even if you find a connection, it doesn’t necessarily mean that one caused the other! For example: imagine there was a sudden outbreak of the plague; it’s not always fatal, but many people who contract it will die. Public healthcare expenditure might go up. Life expectancy drops right down. That doesn’t mean that your healthcare system has suddenly become less efficient! You always have to be *REALLY* careful about the conclusions you draw from this kind of data... but it can still be interesting to calculate the figures.

There are many more questions that could be answered using this data. Many of them relate closely to current policy debates. For example, if my country were debating its healthcare spending right now, I could use this data to explore how spending in my country has changed over time, and begin to understand how my country compares to others.

**Formulas** So let’s dive in. The data we have is not entirely complete. At the moment, healthcare expenditure is only shown as a percentage of *GDP*. In order to compare total expenditure in different countries, we need to have this figure in US Dollars (USD).

To calculate this, let’s introduce you to spreadsheet formulas.

Formulas are what helped spreadsheets become an important tool. But how do they work? Let’s find out by playing with them...

**Tip:** Whenever you download a dataset, the very first thing you should do is to make a copy of it. Any changes you should make should be done in this copy - the original data should remain pure and untouched! This means that you can go back and check it at any time. It’s also good practice to note where you got your data from, when and how it was retrieved.

Once you have your own copy of the data (try adding ‘working copy’ or similar after the original name), create a new sheet within your spreadsheet. This is for you to mess around with whilst you learn about formulae.

Now move across to the “Total fruits sold” column. Start in the first row. It’s time to write a formula...

**Walkthrough: Using spreadsheets to add values.** Using this example data. Let’s calculate the total of fruits sold.

1. Get the data and create a working copy.
2. To start, move to the first row.
3. Each formula in a spreadsheet starts with `=`
4. Enter `=` and select the first cell you want to add. Notice how the cell reference appears in the formula?
5. now type `+` and select the second cell you want to add
6. Press `Enter` or `tab`.
7. The formula disappears and is replaced by the value.
8. Try changing the number in one of the original cells (apples or plums) you should see the value in total update automatically.
9. You can type each formula individually, but it also possible to cut and paste or drag formulas across a range of cells.
10. Copy the formula you have just written (using `ctrl + c`) and paste it into the cell below (using `ctrl + v`), you will get the sum of the two numbers on the row below.

11. Alternatively click on the lower right corner of the cell (the blue square), and drag the formula down to the bottom of the column. Watch the ‘total’ column update. Feels like magic!

**Task:** Create a formula to calculate the total amount of apples and plums sold during the week.

**Did you add all of the cells up manually?:** That’s a lot of clicking - for big spreadsheets, adding each cell manually could take a long time. Take a look at the “spreadsheet formulae” section in the Handbook - can you see a way add a range of cells or entire columns simply?

**Where Next?** Once you’ve got the hang of building a basic formula - the sky is your limit! The School of Data Handbook will additionally walk you through:

- Multiplication using spreadsheets
- Division using spreadsheets
- Copying formulae sideways
- Calculating minimum and maximum values
- Dealing with empty cells in your data (complex formulae). This stage uses Boolean logic.

You may need to refer to these chapters to complete the following challenges.

**Multiplication and division challenge** **Task:** Using the data from the World Bank (if you don’t have it already, download it here.). In the data we have figures for healthcare only as a % of GDP. Calculate the full amount of private health expenditure in Afghanistan in 2001 in USD. If your percentages are rusty - check out the formulae section in the Handbook.

**Task:** Still using the World Bank Data. Find out how much money (USD) is spent on healthcare per person in Albania in 2000.

**Task:** What is the formula for healthcare expenditure per capita? Can you modify it so it’s only calculated when both values are present (i.e. neither cell is blank)?

**Summary & Further Reading** In this module we had an in depth view on analysis. We explored our dataset looking at the range of data. We further took a leap into conditional formulas to handle missing values and developed a quite complex formula step by step. Finally we touched on the subject of normalizing data to compare entities.

1. Google Spreadsheets Function List
2. Introduction to Boolean Logic at the Wikiversity

Take the following quiz to check your analysis skills.

**From Data to Diagrams: An introduction to plots and charts**

**Introduction** The last tutorials was all text and grey, so let’s add some glitter to the world of data: Data Visualization. Data visualization is not just about making what you found look good - often it is a way of gaining insight into the data. We just understand graphical information on a better level than we understand numbers and tables. Look at the example below: How long does it take to see the trend in the table, how long in the chart?

Data visualization is a great skill and if done right has great value. If done incorrectly, you will lead people astray and plant wrong ideas in their heads. Remember: With great power comes great responsibility.
Data Wrangling Handbook, Release 0.1

In this tutorial we have two missions:

- To understand which type of chart is most appropriate to present your data
- To learn the basic workflow for inserting basic charts into a spreadsheet with Google Docs.

For this tutorial you will need to have the share settings on your spreadsheet set to “public on the web” - otherwise some of the things we cover won’t work. Do so by changing the settings with the blue share button on the top right.

In case you haven’t completed the last module the spreadsheet we are working with is here.

How to present data? We have largely quantitative data in our dataset. The question we have to ask ourselves is: Do we compare one entity over time, multiple entities with each other or do we want to know how two variables interact? Depending on this we choose different presentation formats.

<table>
<thead>
<tr>
<th>What we want to do</th>
<th>Presentation chosen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compare values from different categories</td>
<td>barchart</td>
</tr>
<tr>
<td>Follow value over time (timeseries)</td>
<td>linechart</td>
</tr>
<tr>
<td>Show interaction between two values</td>
<td>scatterplot</td>
</tr>
<tr>
<td>Show data related to geography</td>
<td>map</td>
</tr>
</tbody>
</table>

Presenting quantitative data from different categories - Bar/columncharts A barchart is one of the most commonly used forms of presenting quantitative data. It is simple to create and to understand. It is best used when comparing data from different categories: e.g. public healthcare expenditure in the top 10 countries - and as 11th column your country. A typical columnchart looks like this:

Reading barcharts is simple: We usually have a few values - ordered as categories on the x or y axis (for column and barcharts respectively) in our example it’s the countries. Then we have the values expressed as bars (horizontal) or columns (vertical). The extent of the bars is the value.

As simple as it is there are a few rules to keep in mind:

- Don’t overload barcharts Although you can do multiple colours and pack two categories in there, if it’s too many categories it becomes confusing.
- Always label your axes whoever is looking at your graphs needs to know what the units are they are looking at.
- Start your values at 0. Most spreadsheet tools will automatically adjust the range: undo this and set it to 0 - this shows contrast in an appropriate scale! We’ll show you why this is important in the next module.

Walkthrough: Create a column chart for the top 10 countries. So let’s create a column chart from our dataset. It’s not really good style to have too many different columns in there: the chart becomes very hard to read. So what we will do is to limit ourselves to the 10 countries with the highest healthcare expenditure. This is an arbitrary cutoff and you can look at all the countries as well. Doing so might help you discover something that’s hidden in the data.

1. To do so, filter the World Bank dataset for a single year (e.g. 2009).
2. Sort the filtered world bank data set by the column “Health care expenditure total per person (US$)” one of the columns we created in the last challenge. Remember to select the whole sheet - not just the column you’re sorting.
3. Select the top 10 countries (the first 11 rows including the header row) and copy it to another sheet. (For this press ctrl + c for copy and then insert a new sheet, press ctrl + v in the new sheet to paste).
4. Because of the way google docs works, we now have to bring the data columns we are interested in next to the column with the country names (column A).
5. Click on the grey label to select it. Release the mouse then click and drag it until it is in position. Your column A should now be Country Names, Column B should be “healthcare expenditure per person total US$”. Your sheet should look like this:

6. Now select the first two columns and then open chart... from the insert menu.

7. One of the suggested charts should be a column chart

8. Click on it and you will see a preview. Did you note the range on the y axis?

9. It starts with 4000 so it looks like Belgium is only spending a fraction of Luxembourg’s spending on healthcare - let’s change this.

10. Open the Customize tab and scroll down to ”Axis”. Now select ”Left Vertical” from the drop down.

11. See the max and min boxes? Just enter 0 into the min and the range will start at 0. This way the contrast between the countries looks more realistic.

12. Play around with the customizing settings. Try to remove and position the legend, change the colour of your bars etc.

13. When you are done, click on Insert and your chart will be there.

14. If you click on the chart you can move it around. Notice the triangle up right? It’s the option menu. Select Edit chart to change the settings of the chart. Can you change it to a bar chart?

**Task:** Create a column chart with other data from the World Bank sheet.

**So now you know how to create a column chart - feel free to experiment with other types of chart and use the recipes in the Handbook to guide you.** The following sections deal with when to pick a particular type of chart and what data it is suitable for. We cover the most common charts: line charts, choropleth maps and scatterplots. For all of these, you can find an accompanying howto recipe in the handbook.

**Presenting data from categories over time - linecharts** Sometimes you do not only have categories: e.g. countries, but you have values over time. This is where line charts are quite handy. A line chart looks like:

On the y axis we still have our values on the x axis we have the time measured. This graph works best if the time interval between the measurements is equal (Of course line charts are not limited to timeseries). Again it’s important, when comparing multiple categories, to start your y axis with 0. Only when displaying a single line it’s ok to start somewhere in between - but give a relation - say where your graph starts and where it ends.

**Task:** Compare Luxembourg to the other top spending countries - create a line chart with the different countries on one chart.

**Showing geographical data - mapping** In our case we do not only have numerical data but we also have numerical data that is linked to geographical places. This calls for a map! Whenever you have a large number of countries or regions, displaying data on a map helps. If you have countries or regions you usually create a choropleth map. This special type of map displays values for a specific region as colours on that region. An example of a choropleth map from our data is shown below:

The map shows health care expenditure in % of GDP. It allows us to discover find interesting aspects of our dataset. E.g. Western Europe is spending more on healthcare in %GDP than eastern Europe and Liberia spends more than any other state in Africa.

Some things to be aware of when using choropleth maps:

- One shortcoming of choropleth maps are the fact that bigger regions or countries attract most attention, so smaller regions may get lost.
• Pay attention to colour-scales. The standard red-green colour scale is not very well suited for a variety of reasons such as making it difficult for colour-blind observers (Read more about this in Gregor Aisch’s post in the Further Reading section). Single hued colour scales are in most cases easier to guess. If your range of values becomes too big it will be hard to single out things

**Task:** Try another set of data on a choropleth. How does it work?

**Researching interaction between variables - scatterplots** What if we are interested not in a single variable but in how different variables depend on each other? Well in this case we have scatterplots - good for looking at interaction between two variables.

Look at the sample scatterplot above: we have one numerical value on the X and another numerical value on the Y axis. The dots are one data point. This plot has certain shortcomings as well: The dots overlap and thus if there are a lot of dots you don’t really see where they are. This could be solved by adding transparency or by selecting a specific range to show. Nevertheless one trend becomes clear: Above a certain life expectancy, health care costs suddenly increase dramatically. Also notice the three single dots on the lower left? Interesting outliers - we’ll look at them in a later module.

**Task:** Make a scatterplot comparing other data in the dataset. Does it work? Issues, problems, interesting findings?

**Summary** In this tutorial we covered basics of data visualization. We discussed common basic plots and created them. In the next tutorial we will discuss some pitfalls to avoid when handling and interpreting data.

**Further reading**
- “Doing the Line Charts Right” by Gregor Aisch
- Also by Gregor Aisch: Say Goodbye to Red-Green Color Scales

**Look Out!: Common Misconceptions and how to avoid them.**

**Introduction** Do you know the popular phrase: “There are three kinds of lies: lies, damned lies and statistics”? It illustrates the common distrust of numerical data and the way it’s displayed. And it has some truth: for too long, graphical displays of numerical data have been used to manipulate people’s understanding of ‘facts’. There is a basic explanation for this. All information is included in raw data - but before raw data is processed, it’s too much for our brains to understand. Any calculation or visualisation - whether that’s as simple as calculating the average or as complex as producing a 3D chart - involves losing a certain amount of data, so that we can take it in. It’s when people lose data that’s really important and then try to make big statements about the whole data set that most mistakes get made. Often what they say is ‘true’, but it doesn’t give the full story.

In this tutorial we will talk about common misconceptions and pitfalls when people start analysing and visualising. Only if you know the common errors can you avoid making them in your own work and falling for them when they are mistakenly cited in the work of others.

**The average trap** Have you ever read a sentence like: “The average european drinks 1 litre of beer per day”? Did you ask yourself who this mysterious “average european” was and where you could meet him? Bad news: you can’t. He or she doesn’t exist. In some countries, people drink more wine than beer. How about people who don’t drink alcohol at all? And children? Do they drink 1 litre per day too? Clearly this statement is misleading. So how did this number come together?

People who make these kind of claims usually get hold of a large number: e.g. every year 109 billion liters of beer is consumed in Europe. They then simply divide that figure by the number of days per year and the total population of Europe, and then blare out the exciting news. We did the same thing two modules ago when we divided healthcare
expenditure by population. Does this mean that all people spend that much money? No. It means that some spend less and some spend more - what we did was to find the average. The average makes a lot of sense - if data is normally distributed. Normal distribution is the classic bell shaped curve.

The image above shows three different normal distributions. They all have the same average. And yet they are clearly different. What the average doesn’t tell you is the range of data.

Most of the time we do not deal with normal distributions either: take e.g. income. The average income (something frequently reported) would suggest that half of the people would earn less and half of them would earn more than the average. This is wrong. In most countries, many more people earn below the average salary than above it. How? Incomes are not normally distributed. They show a peak around a certain level and then have a long tail towards large salaries.

The chart shows actual income distribution in US$ for households up to 200,000 US$ Income from the 2011 census. You can see a large number of households have incomes around 15,000-65,000 US$, but we have a long tail skewing the average up.

If the average income rises, it could be because most of the people are earning more. But it could also be that a few people in the top income group are earning way more - both would move the average.

**Task:** If you need some figures to help you think of this, try the following:

Imagine 10 people. One earns 1€, one earns 2€, one earns 3€... up to 10€. Work out the average salary.

Now add 1€ to each of their salaries (2€, 3€,...11€). What is the average?

Now go back to the original salaries (1€, 2€, 3€ etc) and add 10€ only to the very top salary (so you have 1€, 2€, 3€... 9€, 20€). What’s the average now?

Economists recognise this and have added another value. The “GINI-Coefficient” tells you something about the distribution of income. The “GINI-Coefficient” is a little complicated to calculate and beyond the scope of this basic introduction. However, it is worth knowing it exists. A lot of information gets lost when we only calculate an average. Keep your eyes peeled as you read the news and browse online.

**Task:** Can you spot examples of where the use of the average is problematic?

**More than just your average...** So if we’re not to use the average - what should we use? There are various other measures which can be used to give a simple mean figure some more context.

- Combine the average figure with the range; e.g. say range 20-5000 with an average of 50. Take our beer example: it would be slightly better to say 0-5 litres a day with an average of 1 litre.
- Use the median: the median is the value right in the middle where 50% of values are above and 50% of values are below. For the median income it holds true that 50% of people earn less and 50% of people earn more.
- Use quartiles or percentiles: Quartiles are like the median but for 25,50 and 75%. Percentiles are the same but for varying percent ranges (usually 10% steps.) This gives us way more information than the average - it also tells us something about the distribution of data (e.g. do 1% of the people really hold 80% of the wealth?)

**Size matters** In data visualization, size actually matters. Look at the two column charts below:

Imagine the headlines for these two graphs. For the graph on the left, you might read “Health Expenditure in Finland Explodes!”. The graph on the right might come under the headline “Health Expenditure in Finland remains mainly stable”. Now look at the data. It’s the same data presented in two different (incorrect) ways.

**Task:** Can you spot why the data is misleading?

In the graph on the left, the data doesn’t start at $0, but somewhere around $3000. This makes the differences appear proportionally much larger - for example, expenditure from 2001-2002 appears to have tripled, at least! In reality, this wasn’t the case. The square aspect ratio (the graph is the same height as width) of the graph further aggravates the effect.
The graph on the right starts with $0 but has a range up to $30,000, even though our data only ranges to $9000. This is more accurate than the graph on the left, but is still confusing. No wonder people think of statistics as lies if they are used to deceive people about data.

This example illustrates how important it is to visualize your data properly. Here are some simple rules:

- Always use a range that is appropriate to your data
- Note it properly on the respective axis!
- The changes in size we see in a chart should actually reflect the change of size in your data. So if your data shows B is 2 times A, then B should be 2 times bigger in your visualization.

The simple “reflect the size” rule becomes even more difficult in 2 dimensions, when you have to worry about the total area. At one point, news outlets started to replace columns with pictures, and then continue to scale the dimensions of pictures up in the old way. The problem? If you adjust the height to reflect the change and the width automatically increases with it, the area increases even more and will become completely wrong! Confused? Look at these bubbles:

**Task:** We want to show that B is double the size of A. Which representation is correct? Why?

**Answer:** The diagram on the right.

Remember the formula for calculating the area of a circle? (Area = \( \pi r^2 \)) If this doesn’t look familiar, see [here](#). In the left hand diagram, the radius of A (r) was doubled. This means that the total area goes up by a scale factor of four! This is wrong. If B is to represent a number twice the size of A, we need the area of B to be double the area of A. To correctly calculate this, we need to adjust the length of the radius by 2. This gives us a realistic change in size.

**Time will tell?** Time lines are also critical when displaying data. Look at the chart below:

A clear stable increase in health care costs since 2002? Not quite. Notice how before 2004, there are 1 year steps. After, there is a gap between 2004 and 2007, and 2007 and 2009. This presentation makes us believe that healthcare expenditure increases continuously at the same rate since 2002 - but actually it doesn’t. So if you deal with time lines: make sure that the spacing between the data points are correct! Only then will you be able to see the trends correctly.

![Figure 1.2: by XKCD](#)

**Correlation is not causation** This misunderstanding is so common and well known that it has its own [wikipedia article](#). There is nothing more to say about this. Simply because two data points show changes that can be correlated, it doesn’t mean that one causes the other.

**Context, context, context** One thing incredibly important for data is context: A number or quality doesn’t mean a thing if you don’t give context. So explain what you are showing - explain how it is read, explain where the data comes from and explain what you did with it. If you give the proper context the conclusion should come right out of the data.

**Percent versus Percentage points change** This is a common pitfall for many of us. If a value changes from 5% to 10% how many percent is the change?

If you answered 5% - I’m afraid you’re wrong! The answer is 100% (10% is 200% of 5%). It’s a change in 5 percentage points. So take care the next time people try to report on elections, surveys and the like - can you spot their errors?

Need a refresher on how to calculate percentage change? Check out the “Maths is Fun” [page](#) on it.
**Catching the thief - sensitivity and large numbers**  Imagine, you are a shop owner and you just installed and electronic theft detection system. The system has a 99% accuracy of detecting theft. The alarm goes off, how likely is it, that the person who just passed is a thief?

It’s tempting to answer that there is a 99% chance that this person stole something. But actually, that isn’t necessarily the case.

In your store you’ll have honest customers and shoplifters. However, the honest customers outnumber the thieves:: there are 10,000 honest customers and just 1 thief. If all of them pass in front of your alarm, the alarm will sound 101 times. 1% of the time, it will mistakenly identify a honest customer as a thief - so it will sound 100 times. 99% of the time, it will correctly recognise that a shoplifter is a shoplifter. So it will probably sound once when your thief does walk past. But of the 101 times it sounds, only 1 time will there actually be a shoplifter in your store. So the chance that a person is *actually* a thief when it sounds is just below 1% (0.99%, if you want to be picky).

Overestimating the probability if something is reported positive in such a scenario is called the base rate fallacy. This explains why airport searches and other methods of mass screening always will turn up lots of false positives.

**Summary**  In this module we reviewed a few common mistakes made when presenting data. When using data as a tool to tell stories or to communicate our issues and results. While we need simplification to understand what the data means - doing it wrong will mislead us. When we present graphical evidence: try to stay true to the data itself. If possible: don’t only release your analysis: release the raw data as well!

**Tell me a story: Working out what’s interesting in your data**

**Introduction**  Data alone is not very accessible. However it is a great fundament to build on. To create information from data it needs to be made tangible. Telling a story with the data is the most straightforward way to do this.

To tell a story with you data you need to figure out certain questions. Why would someone be interested in your story? Who is the someone? How does that someone connect or interact with you data?

**The process**  The process of telling a story with data is very similar to this track. It includes

1. Finding Data - find the data that is suitable to answer your question
2. Wrangle the Data - bring it to a format that is useable
3. Merge Datasets - Bring different datasets together
4. Filter and sort the Data - Pick the data that is interesting
5. Analyze Data - Is there something to it?
6. Visualize Data - If there is something interesting in the data how can we best showcase it to others?

**Finding a Story in Data**  Sometimes you will start out to explore a dataset with a given question in mind. Sometimes you start with a dataset and want to find a story hidden in it. In both cases visualizing the data will be helpful to find the interesting parts. A good way to discover stories is to have interactive visualizations. Live bubble charts are great to do so -since we can compare multiple values at once.

Making the data relevant and close to issues that people care about is one of the hardest things to get right, and the best way to learn is to look for inspiration from people who are really good at it. Here’s a small list to start you thinking:

- **Making a policy point with impact**: Hans Rosling has made a career out of his theatrical way of bringing data about Global Development Data to life. His are now amongst the most watched TED talks. In the recipes section, you will learn how to make interactive bubble diagrams such as the ones he displays here.
• **Data Driven Journalism**: If you want to get the hang of data storytelling, take a lesson from the people it comes naturally to: journalists. The Data Journalism Handbook highlights some of the best data stories and details how and why they were produced in the words of leading Data Journalists from around the world.

• **Storytelling for campaigns**: Tactical Technology Collective have produced an excellent guide to how to target visual information to get your message across and correctly target your audience. Drawing By Numbers teaches people to identify how much detail a reader desires or requires, so that people are neither overwhelmed or bored by the amount of data they are given. They group information design around three basic principles. Your approach should be targeted at whether your audience wants to: *Get the idea, Get the picture or Get the detail*. They also give some great examples of visual campaigns with real impact.

• **Making data personal**: Who are you trying to connect to when you present your data? Will the average citizen be able to relate to spending numbers of the UK government in billions, or would it be more helpful to break them down into numbers that they can relate to and mean something to them? For example, Where Does My Money Go? shows the user, on a daily basis, how much tax they contribute to various spending areas and presents the user with numbers they can better relate to.

**Telling the story** Once you’re through the steps: How do you frame your data? How do you provide the context needed? What format are you going to choose? It could be an article, a blog post, an infographic, or an interactive website dedicated to just this problem. The formats vary as do the ways to tell your story. So what format you pick also depends on who you are and what you want to tell. Are you with a NGO and want to use the data for a campaign? Are you a journalist and want to use the data with a story? Are you a researcher trying to make sense of a research data? Or just a curious blogger looking for interesting things? You will have different audiences and different means to tell your story. Don’t be afraid to share your work with friends and colleagues early - they can give you great insight on how to improve your presentation and story.

**Task**: What stories can be told from the World Bank data and can you identify additional information or data to tell better stories?

**Publishing your results online** Once you’ve gone to all of the effort of finding the juicy parts in your data - you’re ready to get your results online. Many services allow easy ways to embed visualisations & data, such as iframes which you can copy and paste into a blog or website. However, if you are not given an easy way to get your material online, we’ve put together a quick recipe to help you publish your results directly as a webpage.

**Summary** Throughout the Data Fundamentals series we started out acquiring and storing a dataset in a spreadsheet, exploring it and calculating new values, visualizing and finally telling a story about it. Of course there is much more to data than we covered in this basic course. You won’t be on your own though the School of Data is here to help. Now go out, look at what others have done and explore data!

**A Gentle Introduction to Data Cleaning**

This School of Data course is a gentle introduction to reducing errors by cleaning data. It was created by the Tactical Technology Collective and gives you a clear overview what can go wrong in spreadsheets and how to fix it (if it does). Take this course if you want to learn why it is important to clean data and how to do it.

**A Gentle introduction to exploring and understanding data**

The second course by Tactical Technology Collective shows how to use pivot tables to explore your data and gain insights quickly.
School of Data Journalism

Getting Stories from Data  Join Steve Doig and Caelainn Barr for an introduction to finding stories in data.

Spending Stories  Join Friedrich Lindenberg and Lucy Chambers at the School of Data Journalism in Perugia. This workshop gives a quick overview of how to use Google Refine for exploring enormous databases of government financial transactions and its sister service, OpenCorporates, for linking up corporate information with spending information.

Making Data Pretty  Excel, Google Charts API, Raphael, Google Fusion Tables and DataWrapper - a whistlestop tour of some of the best off-the-shelf tools out there for visualising data...

Asking for Data  Heather Brooke, Steve Doig and Helen Darbishire describe how they use Freedom of Information Requests to liberate data from government sources.

A Gentle Introduction to Geocoding

Geocoding is the conversion of a human-readable location name into a numeric (or other machine-processable) location such as a longitude and latitude. For example:

London => geocoding => {latitude: -0.81, longitude: 51.745}

Geocoding is a common need when working with data as you may only have human-readable locations (e.g. “London” or a zip code like “12245”) but for a computer to display the data on a map or query it requires one to have actual numerical geographical coordinates.

Aside: in the example just given we’ve the term “London” has been converted to a point with a single Latitude and Longitude. Of course, London (the City in the UK) covers a significant area and so a polygon would be a better representation. However, for most purposes a single point is all we need.

Online geocoding  In theory, to do geocoding we just need a database that lists place names and their corresponding coordinates. Several, such open databases exist including geonames and OpenStreetMap.

However, we don’t want to have to do the lookups ourselves - that would either involve programming or a lot of very tedious scrolling.

As a result, various web services have been built which allow look ups online or over a web API. These services also assist in find the best match for a given name – for a given simple place name such as London there may be several matching locations (e.g. London, UK and London, Ontario) and one needs some way to match and rank these alternatives.

Nominatim - An Open Geocoding Service  There are a variety of Geocoding services. We recommend using one based on open data such as the MapQuest Nominatim service which uses the ‘Open Street Map’ database. This service provides both “human-readable” service (HTML) and a “machine-readable” API (JSON and XML) for automated Geocoding.
Geocoding - The challenge    Right, so now it’s time to get your hands dirty.

1. Pick a dataset with locations you would like to geocode
2. Follow the recipe in the handbook to show you how to geolocate.
3. If you like - go one step further and put your data on a map. There are lots of great services available to do this, Tilemill by Mapbox is very elegant, allowing you to customise your map to your house style - the documentation is also very clear and accessible. Google Fusion Tables also allows you to easily plot points on a map and is a popular choice with many data journalists for it’s ease of getting started.

Once you’ve finished - drop us a link to what you have produced in the comments below - that could just be a full geocoded dataset - or a beautiful map, go as far as you need!

Example - ‘Human-readable HTML’

Example - Machine-readable JSON   (JSON is also human-readable if you have a plugin)

A Gentle Introduction to Extracting Data

[raw]

Making data on the web useful: scraping

Introduction    Many times data is not easily accessible - although it does exist. As much as we wish everything was available in CSV or the format of our choice - most data is published in different forms on the web. What if you want to use the data to combine it with other datasets and explore it independently?

Scraping to the rescue!

Scraping describes the method to extract data hidden in documents - such as Web Pages and PDFs and make it useable for further processing. It is among the most useful skills if you set out to investigate data - and most of the time it’s not especially challenging. For the most simple ways of scraping you don’t even need to know how to write code.

This example relies heavily on Google Chrome for the first part. Some things work well with other browsers, however we will be using one specific browser extension only available on Chrome. If you can’t install Chrome, don’t worry the principles remain similar.

Code-free Scraping in 5 minutes using Google Spreadsheets & Google Chrome    Knowing the structure of a website is the first step towards extracting and using the data. Let’s get our data into a spreadsheet - so we can use it further. An easy way to do this is provided by a special formula in Google Spreadsheets.

Save yourselves hours of time in copy-paste agony with the ImportHTML command in Google Spreadsheets. It really is magic!

Recipes    In order to complete the next challenge, take a look in the Handbook at one of the following recipes:

1. Extracting data from HTML tables.
2. Scraping using the Scraper Extension for Chrome

Both methods are useful for:
• Extracting individual lists or tables from single webpages

The latter can do slightly more complex tasks, such as extracting nested information. Take a look at the recipe for more details.

Neither will work for:
• Extracting data spread across multiple webpages

**Challenge Task:** Find a website with a table and scrape the information from it. Share your result on datahub.io (make sure to tag your dataset with schoolofdata.org)

**Tip** Once you’ve got your table into the spreadsheet, you may want to move it around, or put it in another sheet. Right click the top left cell and select “paste special” - “paste values only”.

**Scraping more than one webpage: Scraperwiki** Note: Before proceeding into full scraping mode, it’s helpful to understand the flesh and bones of what makes up a webpage. Read the Introduction to HTML recipe in the handbook.

Until now we’ve only scraped data from a single webpage. What if there are more? Or you want to scrape complex databases? You’ll need to learn how to program - at least a bit.

It’s beyond the scope of this course to teach how to scrape, our aim here is to help you understand whether it is worth investing your time to learn, and to point you at some useful resources to help you on your way!

**Structure of a scraper** Scrapers are comprised of three core parts:

1. A queue of pages to scrape
2. An area for structured data to be stored, such as a database
3. A downloader and parser that adds URLs to the queue and/or structured information to the database.

Fortunately for you there is a good website for programming scrapers: ScraperWiki.com

ScraperWiki has two main functions: You can write scrapers - which are optionally run regularly and the data is available to everyone visiting - or you can request them to write scrapers for you. The latter costs some money - however it helps to contact the Scraperwiki community (Google Group) someone might get excited about your project and help you!

If you are interested in writing scrapers with Scraperwiki, check out this sample scraper - scraping some data about Parliament. Click View source to see the details. Also check out the Scraperwiki documentation: https://scraperwiki.com/docs/python/

**When should I make the investment to learn how to scrape?** A few reasons (non-exhaustive list!):

1. If you regularly have to extract data where there are numerous tables in one page.
2. If your information is spread across numerous pages.
3. If you want to run the scraper regularly (e.g. if information is released every week or month).
4. If you want things like email alerts if information on a particular webpage changes.

...And you don’t want to pay someone else to do it for you!
Summary: In this course we’ve covered Web scraping and how to extract data from websites. The main function of scraping is to convert data that is semi-structured into structured data and make it easily useable for further processing. While this is a relatively simple task with a bit of programming - for single webpages it is also feasible without any programming at all. We’ve introduced `importHTML` and the Scraper extension for your scraping needs.

Further Reading

- **Scraping for Journalism: A Guide for Collecting Data**: ProPublica Guides
- **Scraping for Journalists (ebook)**: Paul Bradshaw
- **Scrape the Web: Strategies for programming websites that don’t expect it**: Talk from PyCon
- **An Introduction to Compassionate Screen Scraping**: Will Larson

Extracting Data from PDFs It’s happened to all of us, we want some nice, fresh data that we can sort, analyse and visualise and instead, we get a PDF. What a disappointment.

This course will guide you through the main decisions involved in getting data out of PDFs into a format that you can easily use in data projects.

Navigate the slideshow with the arrow keys.

When cut and paste fail, you’ll sometimes need to resort to more powerful means. Below is a little more information on the two of the trickiest paths highlighted by the flowchart above.

**Note: Bad PDFs and worse PDFs** PDFs are not all the same, some are generated from computer programs (best case scenario) but frequently, they are scanned copies of images. Worst case scenario, they are smudged, tea stained, crooked scans of images. In the latter case - your job will be considerably harder. Nevertheless, there are a couple of tips you can take to make your job easier, read on!

(Modified from text contributed by Tim McNamara)

**Path 1: OCR**

**The OCR Pipeline** Creating a system for Optical Character Recognition (OCR) can be challenging. In most circumstances, the Data Science Toolkit will be able to extract text from files that you are looking for.

OCR largely involves creating a conveyor belt of programming tools (but read on and you will discover a couple which don’t) The whole process can include several steps:

- Cleaning the content
- Understanding the layout
- Extracting text fragments from pieces of each page, according to the layout of each page
- Reassembling text fragments into a usable form

**Cleaning the pages** This generally involves removing dark splotches left by scanners, straightening pages and adding contrast between the background and the printed text. One of the best free tools for this is unpaper.
**File type conversion** One thing to note is that many OCR engines only support a small number of input file types. Typically, you will need to convert your images to portable pixmap format (.ppm) files.

In this section, we’ll highlight a few of the options for extracting data or text out of a PDF. We don’t want to reinvent the wheel, with all of these options, you’ll need to read the manuals for the individual piece of software - we aim here to merely serve as a guide to help you choose your weapon!

Without learning to code, the options on this front are unfortunately somewhat limited. Take a look at the following pieces of software:

- **Tabula** - currently causing a lot of buzz and excitement, but you currently need to install your own version, which makes the barrier to entry quite high.
- **ABBYY Finereader** - unfortunately not free but highly regarded by many as a powerful piece of kit for busting data out of its PDF prisons.
- **CometDocs** - an internet based service

Warning - the tools below require you to open your command line to install and run. And some require knowledge of code to use. We mention them here so that you get an idea of what is possible.

The main contenders in the code-based ones are:

- Tesseract OCR
- Ocropus
- GNU Ocrad
- PDF2HTML

**Path 2: Transcription and Microtasking** Besides the projects mentioned in the presentation, there are a few other options.

The open source project, TaskMeUp is designed to allow you to distribute jobs between hundreds of participants. If you have a project that could benefit from being reviewed by human eyes, this may be an option for you.

Alternatively, there are a small number of commercial firms providing this service. The most well known is Amazon’s Mechanical Turk. They providing something of a wholesale service. You may be better off using a service such as Cloudflower or Microtask. Microtask also has the ethical advantage of not providing service below the minimum wage. Instead, they team up with video game sellers to provide in-game rewards.

**Challenge: Free the Budgets**  
**Task:** Find yourselves some PDFs to bust!

For example, there are many PDFs which need your help in the Budget Library of the International Budget Partnership

Remember - once you’ve liberated your data, share it and save someone else the job! Why not upload to the OpenSpending group on the datahub and drop the OpenSpending Mailing List a line to say you have done so, people are always looking for raw data to visualise and explain.

**Working with Budgets and Spending Data**

In this course, we will take you through the steps involved working with budget and spending data. In the process, you’ll learn how to wrangle and clean up some of the most common errors which we see in spending and budget data, as well as doing some dataset gymnastics, such as transposition and cleanup before creating a simple visualisation at the end.

If you don’t have your data yet, or don’t have it in machine-readable format - take a look at the two courses above on extracting data, we start from cleaning up and formatting your data.
What is the difference between budgets and spending?  This section draws on material from the Spending Data Handbook.

Before you start to work with your financial data, it’s important to get a feeling for the two key different types of data you may encounter.

While these terms may have different meanings on a country by country basis, it is possible to separate the types of data into two basic types by looking at the political significance and technical differences between the data. In this section, we look briefly at the two different types of data and what questions can be addressed using them.

Budget Data - Political Details  Budget data is defined as data relating to the broad funding priorities set forth by a government, often highly aggregated or grouped by goals at a particular agency or ministry. For instance, a government may pass a budget which contains elements such as “Allocate $20 million in funding for clean energy grants” or “Allocate $5 billion for space exploration on Mars”. These data are often produced by a parliament or legislature, on an annual or semi-annual basis.

Spending Data - Execution Details  Spending data is defined as data relating to the specific expenditure of funds from the government. This may take the form of a contract, loan, refundable tax credit, pension fund payments, or payments from other retirement assistance programs and government medical insurance programs. In the context of our previous examples, spending data examples might be a $5,000 grant to Johnson’s Wind Farm for providing renewable wind energy, or a contract for $750,000 to Boeing to build Mars rover component parts. Spending data is often transactional in nature, specifying a recipient, amount, and funding agency or ministry. Sometimes, when the payments are to individuals or there are privacy concerns, the data are aggregated by geographic location or fiscal year.

The fiscal data of some governments may blur the lines of these definitions, but the aim is to separate the political documents from the raw output of government activity. It will always be an ultimate goal to link these two datasets, and to allow the public to see if the funding priorities set by one part of the government are being carried out by another part, but this is often impractical in larger governments since definitions of programs and goals can be “fuzzy” and vary from year to year.

Budget data  Using the definitions above, budget data is often comprised of two main portions: revenue and taxation data and planned expenditures. Revenue and spending are two sides of the same coin and thus deserve to be jointly considered when budget data is released by a government. Especially since revenue tends to be aggregated to protect the privacy of individual taxpayers, it makes more sense to view it alongside the budget data. It often appears aggregated by income bracket (for personal taxes) or by industrial classification (for corporate taxes) but does not appear at all in spending data. Therefore, budget data ends up being the only source for determining trends and changes in revenue data.

Somewhat non-intuitively, revenue data itself can include expenditures as well. When a particular entity or economic behaviour would normally be taxed but an exception is written into the law, this is often referred to as a tax expenditure. Tax expenditures are often reported separately from the budget, often in different documents or at a different time. This often stems from the fact that they are released by separate bodies, such as executive agencies or ministries that are responsible for taxation, instead of the legislature (http://internationalbudget.org/wp-content/uploads/Looking-Beyond-the-Budget-2-Tax-Expenditures.pdf).

Budgets as datasets  A growing number of governments make their budget expenditure data available as machine-readable spreadsheets. This is the preferred method for many users, as it is accessible and requires few software skills to get started. Other countries release longer reports that discuss budget priorities as a narrative. Some countries do something in between where they release reports that contain tables, but that are published in PDF and other formats from which the data is difficult to extract.
On the revenue side, the picture is considerably bleaker, as many governments are still entrenched in the mindset of releasing revenue estimates as large reports that are mostly narrative with little easily extractable data. Tax expenditure reports often suffer from these same problems.

Still, some areas that relate to government revenue are beginning to be much better documented and databases are beginning to be established. This includes budget support through development aid, for which data is published under the IATI (http://www.aidtransparency.net/) and OECD DAC CRS (http://stats.oecd.org/Index.aspx?DatasetCode=CRSNEW) schemes. Data about revenues from extractive industries is starting to be covered under the EITI (http://eiti.org/) with the US and various other regions introducing new rules for mandatory and granular disclosure of extractive revenue. Data regarding loans and debt is fairly scattered, with the World Bank providing a positive example (https://finances.worldbank.org/), while other major lenders (such as the IMF) only report highly aggregated figures. An overview of related data sources can be found at the Public Debt Management Network (http://www.publicdebtstats.org/public/Statistics/).

**Connecting revenues and spending**  It is highly desirable to be able to determine the flow of money from revenues to spending. For the most part, many taxes go into a general fund and many expenditures come out of that general fund, making this comparison moot. But in some cases, in many countries, there are taxes on certain behaviours that are used to fund specific items.

For example, a car registration fee might be used to fund the construction of roads and highways. This would be an example of a user fee, where the main users of the government service are funding it directly. Or you might have a tax on cigarettes and alcohol that funds healthcare grants. In this case, the tax is being used to offset the added healthcare expense of individuals taking part in at-risk activities. Allowing citizens to view what activities are taxed in order to pay for other expenditures makes it possible to see when a particular activity is being cross-subsidized or heavily funded by non-beneficiaries. It can also allow them to see when funds are being diverted or misused. This may not always be practical at the country level, as federal governments tend to make much larger use of the general fund than other local governments. Typically, local governments are more comprehensive with regards to releasing budget data by fund. Having granular, fund-level data is what makes this kind of comparison and oversight possible.

**What questions can be answered using budget data?**  Budget expenditure data has an array of different applications, but it’s prime role is to communicate to it’s user broad trends and priorities in government spending. While it can help to have a prose accompaniment, the data itself promotes a more clear-cut interpretation of proposed government spending over political rhetoric. Additionally, it is much easier to communicate budget priorities by economic sector or category than it is at the spending data level. These data also help citizens and CSOs track government spending year over year, provided that the classification of the budget expenditure data stays relatively consistent.

**Spending data**  For most purposes, spending data can be interpreted as transactional or near-transactional data. Rather than communicating the broad spending priorities of the government like budget data should, spending data is intended to convey specific recipients, geographic locations of spending, more detailed categorisation, or even spending by account number.

Spending data is often created at the executive level, as opposed to legislative, and should be more frequently reported than budget data. It can include many different types of expenditures, such as contracts, grants, loan payments, direct payments for income assistance and maintenance, pension payments, employee salaries and benefits, intergovernmental transfers, insurance payments, and more.

Some types of spending data - such as contracts and grants - can be connected to related procurement information (such as the tender documents and contracts) to add more context regarding the individual payments and to get a clearer picture of the goods and services covered under these transactions.

**Opening the checkbook**  In the past five years, there have been a spate of countries and local governments that have opened up spending data, often referred to as “checkbook level” data. These countries include, but are not limited to,
the US (including various state governments), UK, Brazil, India (including some state governments) and many funds of the European Union.

What questions can be answered using spending data? Spending data can be used in several different areas: oversight and accountability, strategic resource deployment by local governments and charities, and economic research. However, it is first and foremost a primary right of citizens to view detailed information about how their tax dollars are spent. Tracking who gets the money and how it's used is how citizens can detect preferential treatment to certain recipients that may be illegal, or if certain political districts might be getting more than their fair share.

It can also help local governments and charities respond to areas of social need without duplicating federal spending that is already occurring in a certain district or going to a particular organization. Lastly, businesses can see where the government is making infrastructure improvements and investments and use that criteria when selecting future sites of business locations. These are only a few examples of the potential uses of spending data. It’s no coincidence that it has ended up in a variety of commercial and non-commercial software products – it has a real, economic value as well as an intangible value as a societal good and anti-corruption measure.

Task: Examine whether budget and / or spending data are available for your country. Note: these may have different names (e.g. Enacted Budget)! “Budget” and “spending” are categories rather than necessarily the names of the documents, policy-folks may actually refer to what we refer to here as spending as budgets (with a particular qualifier).

Save your file online somewhere. You may like to add it to the OpenSpending group on the Datahub - make sure to add all the necessary tags and details so that you (and others) can find it there.

Extra Credit: Read the definition of machine readable in the School of Data glossary. Determine whether your budget or spending data is machine readable. You will need machine readable data in the next stage, so if it is non machine readable you may:

1. If your data is available in a webpage, but there is no download link - scrape it! Take the introduction to scraping course. (Don’t be afraid - you don’t necessarily need to be able to code!)
2. If your data is available in a PDF - take the extracting data from PDFs course. # Note: You may find it easier to submit a Freedom of Information request for machine-readable data. See this example of a successful request for machine-readable data for the EU budget. Don’t forget to ask for any of the supporting documents which help you to understand things like jargon, or how figures were calculated!

Categorization and reference data Adapted from the Spending Data Handbook. One of the most powerful ways of making data more meaningful for analysis is to combine it with reference data and code sheets. Unlike transaction data - such as statistical time series or budget figures - reference data does not describe observations about reality - it merely contains additional details on category schemes, government programmes, persons, companies or geographies mentioned in the data.

For example, in the German federal budget, each line item is identified through an eleven-digit code. This code includes three-digit identifiers for the functional and economic purpose of the allocation. By extending the budget data with the titles and descriptions of each economic and functional taxonomy entry, two additional dimensions become available that enable queries such as the overall pension commitments of the government, or the sum of all programmes with defence functions.

The main groups of reference data that are used with government finance include code sheets, geographic identifiers and identifiers for companies and other organizations:

Classification reference data Reference data are dictionaries for the categorizations included in a financial datasets. They may include descriptions of government programmes, economic, functional or institutional classification schemes, charts of account and many other types of schemes used to classify and allocate expenditure.

Some such schemes are also standardized beyond individual countries, such as the UN’s classification of functions of government (COFOG - http://unstats.un.org/unsd/cr/registry/regest.asp?Cl=4) and the OECD DAC Sector codes
Still, the large majority of governments use their own code sheets to allocate and classify expenditure. In such cases, it is often advisable to request access to the code list versions used internally by government, including revisions over time that may have changed how certain programmes were classified.

A library of reference data that can be re-used across different projects and it is a valuable asset for any organization working with government finance. Sharing such data with others is crucial, as it will help to enable comparable outputs and open up options for future data integration. Existing repositories include the IATI Standard (http://iatistandard.org/) and datahub.io.

**Geographic identifiers**  
Geographic identifiers are used to describe administrative boundaries or specific locations identified in a dataset. While some regional classifications (such as the EU NUTS) are released on the web, there is also an increasing number of open databases which contain geographic names - including geonames.org and the recently developed world.db.

Another related technique is the process of reverse geo-coding: translating a human-readable address into a pair of coordinates. Services like nominatim (http://nominatim.openstreetmap.org/) will not only enable users to generate precise maps of projects in a region, they will also return the responsible administrative boundary for many coordinates. This means that projects which are given by precise address can also be aggregated by state, region or any other geographic unit.

Additionally, many countries have shapefiles of their political and geographic districts available (usually through the census or interior bureaus) that can be imported into custom mapping applications, like TileMill (http://mapbox.com/tilemill/).

**Company and organisational identifiers**  
As you look into spending data that includes recipients outside the government, you’ll find companies which act as suppliers to government, but also other types of entities including charities, associations, foreign governments, political parties and even individuals which act as recipients of direct assistance.

Identifying such entities is notoriously hard, since the only information kept by government is often a simple name (which may not uniquely identify the beneficiary, for example “MS”). While most (if not all) countries maintain company registers which assign some type of unique identifier to a company, these databases are often not accessible in bulk and not used coherently across different parts of government. Alternative identifiers - such as tax identifiers and company IDs from private business information suppliers (such as Dun & Bradstreet in the US) - further complicate this process.

As an alternative, open registries are beginning to compile organisational identifiers in a form that is easy to re-use and thus enables the sharing of databases which have been augmented with such information. OpenCorporates.com (http://opencorporates.com) is a startup that collects information from companies world-wide and provides a convenient API to match datasets with the list of known countries. They offer a service to ‘reconcile’ companies to link information about a company to a company name. This is especially useful when you have an exist spreadsheet or dataset featuring lots of companies. Matching (or reconciling) to legal entities allows you to get more information about the companies (for example the registered address or statutory filings), and makes it easier to match with other datasets or exchange with other organisations.

The IATI project for aid transparency is working towards similar standards for other organisations, such as foreign governments and charities active in the development space.

**Tasks**

1. Find out - does your country use a standard classification for its data such as COFOG?
2. If you have code sheets for your data - try combining them with your main dataset. Take the merging data course if you need a hand.
3. If relevant: Try geocoding your data. For example - does your data include particular projects? Can you find an address for them? If so, you can geocode them and put them on a map! Take a look at the geocoding course if you need more help.
4. If you have spending data, try to answer the question “how much did company X receive from government”. You may need to correct typos and find out answers to questions such as “is company X” the same thing as “company X Ltd” (this is why having organisational identifiers is so important)! If there is company data for your country, use the OpenCorporates reconciliation service to find out more information about the company (on the right hand side of the page on OpenCorporates.com).

Cleaning spending data Often, you will need to clean up the spending data that you receive. Even government data is not perfect.

In this section, we highlight some common issues experience when working with spending data.

In the recipe book - you will find a recipe to help you solve these issues.

Common issues with spending data:

• **Typos** - E.g. some of your cells contain “Rroceeds from global taxes” rather than “Proceeds from global taxes”

• **Inconsistencies** - You may be looking to try and find out how much money does company X receive from government. However in your data, sometimes the company is entered variously as “Fuzzy Robot Llama Exporters”, “Fuzzy Robot Llama Exporters Ltd”, “FRLE Ltd” - you need a way to tell that these are all the same.

• **Blank columns, rows and cells** - When you are summing up values it is very important to know what is a genuine zero and what is a blank due to the absence of data.

• **Human Friendly Formatting** - such as pseudo rows and things being in horizontal rows when you need them in verticals - your computer probably needs the computer to be laid out somewhat consistently if it is to be able to process it.

• **Multiple types of information contained in a single column.** For example:

Actually, it would be more helpful to have a column just to have one column with just whether a transaction was revenue or expenditure and a second to allow you to filter by revenue / expenditure type.

• **Whitespace** - You don’t see it, but it causes big problems in datasets. To many databases, the two transactions below would be treated differently due to the extra space at the end:

So if you ever filtered or searched for just “Fiscal revenue” [no space at the end] - you would actually be omitting the latter case - possibly leading to incorrect conclusions. You can remove the nasty white space quite easily.

Normalizing data Data that comes from the government is often generated across multiple departments by hand. This can result in inconsistencies in what kinds of values or formats are used to describe the same meaning. Normalizing values to be consistent across a dataset is therefore a common activity.

**Step 1: Find all distinct values** First, you want to start by finding all of the distinct ranges of values for the different columns in your dataset. You can accomplish this by using a database query language (such as SQL’s DISTINCT), or by simply using the ‘filter’ property on a spreadsheet program.

For example, if you have a spreadsheet with contracting data, and one column is ‘Competed?’, you would expect the values to be ‘yes’ or ‘no’. But if this spreadsheet is an amalgam of spreadsheet data from multiple users and departments, your values could vary among the following: ‘Y’, ‘YES’, ‘yes’, 1, ‘True’, ‘T’, ‘t’, ‘N’, ‘NO’, ‘no’, 0, ‘False’, ‘F’, ‘f’, etc. Limiting all of these potential values to two clear options will make it easier to analyse the data, and also easier for those who follow in your footsteps.

**Step 2: Sanity Check** Especially with financial data, numbers can be formatted several different ways. For example, are your negative values represented with a ‘-’ or placed inside ‘( )’ or possibly even highlighted in red? Not all of these values will be easily read by a computer program (especially the color), so you’ll want to pick something clear and consistent to convert all your negative values to (probably the negative sign).
Is all your numerical data measured out in ones or is abbreviated in thousands? Especially with budget data, order of magnitude errors are not uncommon when one department thinks they’re reporting in thousands or millions by default but others expand their data all the way to the ones place. Are some values in scientific notation (e.g. 10e3938)? Make sure all your values are consistent, otherwise your analysis could contain serious errors.

A column of data requiring name normalization:

Challenge: Wrangle the data  Task:
Read the recipe from the handbook walking you through cleaning up a spending dataset. Take the sample dataset and replicate the steps there.

Remember - once you’ve cleaned your data, share it and save someone else the job! Why not upload to the Open-Spending group on the datapub and drop the OpenSpending Mailing List a line to say you have done so, people are always looking for raw data to visualise and explain.

Extra Credit:
Take a dataset from your own country and clean it up ready to go into a database.

Choosing an audience and classifying your data  Depending on who the audience is for your data, you may want to think about presenting or delivering your data differently.

The Spending Data Handbook covers some important considerations about targeting your data project.

In this chapter, we cover a data conversion technique that the Open Knowledge Foundation employ regularly in order to make data more accessible and (if caution is applied) comparable between countries.

Grouping budgets and spending by topics people care about  Functional classifications, for those of you who don’t regularly ‘sail the wide accounting seas’ tell you things like what general area of spending we are talking about “health”, “education”, “defence”, which is often more interesting from the perspective of the citizen user than e.g. which ministry group it was spent by.

Some governments already classify their data by functional classifications, but you can also group them. No idea what we are talking about or why you should care? Read on:

Why Functional Classifications?  Simply speaking, many users of data want to know what government spent money on, rather than who spent it, who received it. People (I’m talking about the general public here) generally care about services - not bank transfers.

You don’t have to make these classifications up from scratch, there are internationally recognised systems of these. For example, the stylishly named Classifications of Functions of Government (COFOG, for short) - is how the government already publishes its data in the UK. This, with a few amendments to translate the terms from political jargon into terms that people could identify with - was the system used to make the budget understandable in Where Does My Money Go?

For other projects which we’ve done e.g. Cameroon.OpenSpending.org we’ve used a COFOG-esque mapping. Why ‘esque’? Firstly, the government didn’t publish their data classified like this, so we had to group it ourself. Secondly, we were aiming here for a functional classification which worked when you visualised it, if we’d used COFOG exactly for Cameroon’s specific case, we’d have ended up with a huge bubble for general public services which would have made all the others really small, so you wouldn’t be able to see the difference in size. So we modified the set of top-level items to make it easier to see smaller distinctions.
What fits? The other thing to bear in mind, particularly if you are planning on visualising your data, is how many things you can fit on your visualisation. You should consider two factors:

1. How many categories will fit spatially on your visualisation?
2. How many categories can people take in? Lots of categories can be overwhelming for the observer of a visualisation. For Where Does My Money Go? The fact that there were only 10 or so top level items was one of the reasons we were able to use the COFOG classification.

Compare the visualisations below:

International Comparisons While we’re always warning people about making comparisons between countries (data not being collected in the right way etc lalala), these classifications using COFOG are quite often used to make international comparisons. OECD do it regularly, so it’s probably one of the less-evil ways to do it, in case you’re interested in that type of thing.

De-jargonising COFOG Let’s face it, the terminology used by the government is often not the most appealing, or illustrative from the point of view of the user. Hence, for the Where Does My Money Go Project, we specifically de-jargonised it, and translated the terms into friendly forms that we felt were more accessible to the average user. For example: ‘Executive and legislative organs, financial and fiscal affairs, external affairs’ became ‘Top level government’. You can take a look at how we mapped them on to one-another in this Google Doc.

How to map your budget into COFOG classifications: Basically, if your government doesn’t do this for you - you’ll always have to use your best judgements, someone may have made a different call, and may well disagree with the way you’ve done what you’ve done. But as long as you document your practices, anyone will be able to pick up anything they don’t agree with and produce a different model.

1. Make a codesheet and align your functional groups to the things you want to go under that umbrella term.
2. Do a dataset cross using Google Refine or use HLOOKUP in Excel. Dataset merging will allow you to match information from different data sources or spreadsheets, without merging them, so the original data remains available.

Task:
Find budgets or spending for your country. Take a look at Where Does My Money Go? if you need inspiration. You are now ready to produce a visualisation with your data!

Recipes

Sometimes you’re not hungry for a background heavy lunch but like to snack on something fast: Here are some recipes! This section contains small snippets that will help you in the process of data wrangling. They might be small useful tips or full blown tutorials on tools or topics. Come in and find out!

How to find data

If you are looking for further inspiration on how to find data beyond the ways highlighted in the course on finding data, read on!

Section author: Tim McNamara <tim.mcnamara@okfn.org>
Edited directories  One of the largest directories of open data repositories is provided by the Open Access Directory. Its collection is mostly focused on scientific or research data and is curated by topic area. Topics covered in the directory include archaeology, astronomy, biology, chemistry, computer science, energy, environmental sciences, earth sciences, linguistics, marine sciences, medicine, physics and social sciences.

CKAN is a directory that largely works through wiki-like edits. Some of the benefits of CKAN are that it has well developed client libraries that enable you to programatically access information about each of the datasets within its directory. For example, it is easy to ask it to tell you which datasets have been released into the public domain.

Quora has actually become a great source of information about where to find data on specific topic areas. It has several questions related to this topic which are being continually updated. Some examples include:

- What are some free, public data sets?
- Where can I get large datasets open to the public?

In Data Fundamental, we address the question of how healthcare spending affects life expectancy around the world. This is one of the answers we can find looking at data from the World Bank.

Walkthrough: Downloading Data from the World Bank

2. Select Data Catalog from the menu on the top.
3. In the long list on the bottom find “World Development Indicators”
4. Click on the blue databank button next to it.
5. You’ll find a very different site: The Databank - The databank is an interface to the World Bank database. You can select what data you want to see from which countries for what period of time.
6. First select the countries. We’re interested in all the countries click on select all in the country view and then on next.
7. Now you’ll see a long list of data series you can export. We’ll need a few of them.
8. First we are interested in healthcare expenditure so type “Health” in the little search box on the top of the list and click Go
9. Select “Health expenditure, private (% GDP)”, “Health expenditure, public (% GDP)” and “Health expenditure, total (% GDP)”. And click on Select
10. Since the expenditure is in % of GDP we’ll need to get the GDP as well. Since we want to compare countries directly we’ll need GDP in US$. To do this type GDP into the search box and find the entry “GDP (current US$)”
11. If we want to see how healthcare expenditure affects the life expectancy we need to add life expectancy to the data.
12. Now let’s add one more thing: Population - like this we can calculate how much is spent by and on an average person. Search for “Population” and select “Population, total”.
13. Bring GDP and Population to the top with the arrows on the side of the list, your selection should now look like this:
14. Click on Next to select the years we are interested in.
15. To keep things simple, select the years 2000-2012 (you can do multiple selections by either pressing ctrl or shift). And click next,
16. You’ll see an overview screen now on the top left there is a rough layout of how your downloaded file will look like. You’ll see “time” in the columns bit and “series” in the rows bit - this influences how the spreadsheet will look like.

17. While this might be great for some people: The data is a lot easier to handle if all of our “series” are in columns and the years are different rows. So let’s change this.

18. Simply drag the “time” from columns to “rows” and the “series” from rows to columns

19. Your arranged organization diagram should look like this:

20. Now let’s go and Export

21. If you click on the Export button a pop up will appear asking you for the format. Select CSV.

22. You will then be able to download a file - store and name it in a folder so you remember where it comes from and what it is for.

Liberating HTML Data Tables

Section author: Tony Hirst (psychemedia on Twitter)

It’s not uncommon to see small data sets published on the web using an HTML table element. If you have a quick click around Wikipedia, you’re likely to find a wide variety of examples. Some sites will use Javascript libraries to enhance the presentation or usability of a table, for example, by making columns sortable; but most of the time, we are faced with a flat HTML table, and the data locked in it.

In this section, we look at some quick tricks for liberating data from HTML tables on public webpages and turning them into something more useful.

Screenscraping HTML Tables Using Google Spreadsheets

The Google spreadsheet formula:

\[
=\text{importHTML("","table",N)}
\]

will scrape a table from an HTML web page into a Google spreadsheet. The URL of the target web page, and the target table element both need to be in double quotes. The number N identifies the N’th table in the page (counting starts at 1) as the target table for data scraping.

So for example, have a look at the following Wikipedia page – List of largest United Kingdom settlements by population (found using a search on Wikipedia for UK city population):

Grab the URL, fire up a new Google spreadsheet, and start to enter the formula \(=\text{importHTML}\) into one of the cells:

Autocompletion works a treat, so finish off the expression and add in the URL and table number:

\[=\text{importHTML("http://en.wikipedia.org/wiki/List_of_largest_United_Kingdom_settlements_by_population","table",2)}\]

As if by magic, a data table appears in the spreadsheet, pulled in directly from the Wikipedia page:

If the data in the HTML table is updated, the data in the spreadsheet will also be updated when you refresh or call the spreadsheet page.

Scraping websites using the Scraper extension for Chrome

If you are using Google Chrome there is a browser extension for scraping web pages. It’s called “Scraper” and it is easy to use. It will help you scrape a website’s content and upload the results to google docs.
Walkthrough: Scraping a website with the Scraper extension

1. Open Google Chrome and click on Chrome Web Store
2. Search for “Scraper” in extensions
3. The first search result is the “Scraper” extension
4. Click the add to chrome button.
5. Now let’s go back to the listing of UK MPs
6. Open http://www.parliament.uk/mps-lords-and-offices/mps/
7. Now mark the entry for one MP
8. Right click and select “scrape similar...”
9. A new window will appear - the scraper console
10. In the scraper console you will see the scraped content
11. Click on “Save to Google Docs...” to save the scraped content as a Google Spreadsheet.

Walkthrough: extended scraping with the Scraper extension  

Note: Before beginning this recipe - you may find it useful to understand a bit about HTML. Read our HTML primer.

Easy wasn’t it? Now let’s do something a little more complicated. Let’s say we’re interested in the roles a specific actress played. The source for all kinds of data on this is the IMDB (You can also search on sites like DBpedia or Freebase for this kinds of information; however, we’ll stick to IMDB to show the principle)

1. Let’s say we’re interested in creating a timeline with all the movies the Italian actress Asia Argento ever starred; where do we start?
2. The IMDB has a quite comprehensive archive of actors. Asia Argento’s site is: http://www.imdb.com/name/nm0000782/
3. If you open the page you’ll see all the roles she ever played, together with a title and the year - let’s scrape this information
4. Try to scrape it like we did above
5. You’ll see the list comes out garbled - this is because the list here is structured quite differently.
6. Go to the scraper console. Notice the small box on the upper left, saying XPath?
7. XPath is a query language for HTML and XML.
8. XPath can help you find the elements in the page you’re interested in - all you need to do is find the right element and then write the xpath for it.
9. Now let’s assemble our table.
10. You’ll see that our current Xpath - the one including the whole information is “//div[3]/div[3]/div[2]/div”
11. Xpath is very simple it tells the computer to look at the HTML document and select <div> element number 3, then in this the third one, the second one and then all <div> elements (which if you count down our list, results in exactly where you are right now.
12. However, we’d like to have the data separated out.
13. To do this use the columns part of the scraper console...
14. Let’s find our title first - look at the title using Inspect Element
15. See how the title is within a <b> tag? Let’s add the tag to our xpath.

1.2. An introduction to the data pipeline 35
16. The expression seems to work well: let’s make this our first column
17. In the “Columns” section, change the name of the first column to “title”
18. Now let’s add the XPATH for the title to it
19. The xpaths in the columns section are relative, that means “./b” will select the <b> element
20. add “./b” to the xpath for the title column and click “scrape”
21. See how you only get titles?
22. Now let’s continue for year? Years are within one <span>
23. Create a new column by clicking on the small plus next to your “title” column
24. Now create the “year” column with xpath “./span”
25. Click on scrape and see how the year is added
26. See how easily we got information out of a less structured webpage?

A short introduction to HTML

Getting data from websites might seem a little complicated at first - but rest assured, once you’ve done it a couple of times it will be similar. To extract data from websites we need to peek under the hood and look at the underlying HTML code. Don’t worry you don’t need to understand every detail of it just to be able to do so.

HTML is the acronym for Hypertext Markup Language and is the language used to describe (markup) web pages. It is the underlying language to structure web-page content. HTML itself does not determine the way things look - it only helps to classify and structure content. So let’s peek at some websites.

Walkthrough: Exploring HTML with Google Chrome

1. Open the website listing all MPs for the UK Parliament at http://www.parliament.uk/mps-lords-and-offices/mps/ in Chrome
2. Scroll down to the list of MPs
3. Right click on one of the entries
4. Select “Inspect Element”
5. Chrome will open a second area on the bottom of the page showing the underlying HTML code - focussed on the element you clicked
6. The pointy brackets are the HTML tags.
7. Now move your mouse up and down and notice how chrome tells you which element is which
8. You can expand and collapse certain sections by clicking on the triangles
9. Did you notice something? Every row in the long list of MPs is within one <tr></tr> section. <tr> indicates a table row.
10. The names and the constituency are in <td></td> tags - td indicates table data. So we’re dealing with a table here?
11. If you scroll up the list you’ll notice a <table> element, followed by a <tbody> element - so yes this is a proper HTML table.
12. Go ahead and explore!
HTML is no mystery. If you want to know more about it and how to build webpages with it - visit the School of Webcraft for a gentle introduction.

**Other browsers**  To do the same thing in other browsers, try the following approaches.

- Safari: Preferences > Advanced > Show Develop Menu > Show Web Inspector
- Internet Explorer 7: Install Developer toolbar

**HTML Elements**  Elements are identified by ‘tags’, their name. They can have an inner text and “attributes” (named properties): `<tag attribute="value">text</tag>`

- `<html>` - the whole document
- `<body>` - the human-readable part of the web page
- `<table>` - the frame of a table element
- `<tr>` - a row in a table
- `<td>` - a cell of content inside a row
- `<th>` - a table header cell inside a row

**Python code elements for scraping**

- `name = expression` - assign a name to the output of a computation
- `from lxml import html` - import html component form a “library”
- `doc = html.parse('http://....')` - download and analyze a web page.
- `doc.findall('//tag')` - find all occurrences of a tag in the whole document
- `element.findall('childtag')` - find all othertags within element
- `element.find('highlander')` - find a single highlander within element
- `for name in list-of-things:` - run code on each element of the list, assign the item to name
- `list-name[n]` - get the nth element from a list.
- `scraperwiki.save(unique_keys=[], data={'field': value, 'field2': value})` - see [https://scraperwiki.com/docs/python/python_datastore_guide/](https://scraperwiki.com/docs/python/python_datastore_guide/)

**Task:** Pick a website and look at the HTML code using Inspect Element. Did you find something interesting?

### Scraping - Beyond the Basics

*Section author: Tim McNamara <tim.mcnamara@okfn.org>*

This guide focuses on how you can extract data from web sites and web services. We will go over the various resources at your disposal to find sources which are useful to you.

**Extracting Data**
General tips

• Minimise the pages to scrape. This will save everybody time and resources.
  – Inspect any AJAX fields. AJAX is generally performed by sending JavaScript objects between the server and the web browser. They are easy to parse and are generally very rich.
  – Try looking for a sitemap.xml.
  – Any pages in the robots.txt which disallow access are generally where the bulk of the value lies.
• Run an evented or multi-threaded system. Once you have gained the confidence of building a few scrapers, learn how to optimise performance. Given that you are using lots of external resources, there will be lots of latency involved. This means that your scraper’s performance increases by using asynchronous programming.

Types of scrapers

DOM-based approaches

advantages

• familiar
• relatively computationally efficient

disadvantages

• requires parsing the entire document, which can be difficult with messy content
• prone to breaking when encountering unexpected content
• can be tricky to handle errors
• may require learning a new language, XPath

This is the most common form of scraper. All the data that you are looking to extract is identified by selecting portions from the DOM.

Most modern libraries, such as lxml accept CSS selectors. So, in Python to extract content from the <title> tag, you do something similar to page.cssselect(‘title’)\[0\].text.

**XPath, the XML Path Language, is a fuller way to select elements** from XML and XML-like documents, such as HTML. As with CSS, it uses the structure of the page and tag attributes to be able to select specific elements or groups of elements. XPath expressions can look fairly complex and take some some time to learn.

Template Regular expressions to look for common patterns in the text. One of the easiest template extraction systems is scrapemark. While it is not the most computationally efficient, using template systems requires far less manual work to get going with.

Machine-learning Machine-learning packages work by training a model of example pages, then asking for matching material.

One tool that is very good at removing boilerplate, such as headings from web pages and only leaving the content is called boilerpipe. It is bundled together with the ‘Data Science Toolkit’ and there is an demo of boilerpipe’s capabilities is available.

A scraping framework Let’s demonstrate some of the principles that we have been talking about.

We’ll be creating a scraping framework, called *tbd*:
import bsddb
import pickle
import urllib2
from asynchat import fifo
from dateutil import parser as date_parser
import lxml
import lxml.html

START_URL = 'http://blog.okfn.org/
db = bsddb.hashopen('okfnblog.db')

# UTILITY FUNCTIONS
#

def get_clean_page(url):
    page = get_page(url)
    page = lxml.html.tostring(page)
    page = lxml.html.fromstring(page)
    return page

def get_page(url):
    res = urllib2.urlopen(url)
    page = lxml.html.parse(res)
    page.make_links_absolute()
    return page

def save_post(post):
    save('post_id', post)

def save_tag(tag):
    save('tag-%s' % tag['tag'], tag)

def save_author(author):
    save('author-%s' % author['name'], author)

def save(key, data):
    db[key] = pickle.dumps(data)

def extract_created_at_datetime(post):
    date = post.cssselect('span.entry-date')[0].text
    time = post.cssselect('div.entry-meta a')[0].attrib['title']
    return str(date_parser.parse(date + ' ' + time))

def process_post(url):
    source = get_page(url)
    post = {
        'title': source.cssselect('h1.entry-title')[0].text,
        'author': source.cssselect('span.author a')[0].text,
        'content': source.cssselect('div.entry-content')[0].text_content(),
        'as_html': lxml.html.tostring(source.cssselect('div.entry-content')[0]),
        'created_at': extract_created_at_datetime(source),
        'post_id': source.cssselect('div.post')[0].attrib['id'],
        'tags': [tag.text for tag in source.cssselect('a[rel~tag}')
    }
post['url'] = url
yield save_post, post
yield save_author, dict(name=post['author'])
for tag in post['tags']
    yield save_tag, dict(tag=tag, post_id=post_id, author_name=post['author'])

def process_archive(url):
    archive = get_page(url)
    for post in archive.cssselect('.post .entry-meta a'):
        yield process_post, post.attrib['href']
    previous = archive.cssselect('.nav-previous a')
    if previous:
        yield process_archive, previous[0].attrib['href']

def process_start(url):
    index = get_page(url)
    for anchor in index.cssselect('li#archives-2 a'):
        yield process_archive, anchor.attrib['href']

def main():
    queue = fifo((process_start, START_URL))
    while 1:
        status, data = queue.pop()
        if status != 1:
            break
        func, args = data
        for newjob in func(args):
            queue.push(newjob[0], newjob[1])
        db.sync()

Dealing with JavaScript  JavaScript can be a pain for scrapers. JavaScript is often used to alter the DOM on pages after they have been created. This means that the page you see in an Internet browser is different that the page your scrapers see.

There are a few different approaches to dealing with this process. We will briefly outline them, then go through the easiest option.

Options  There are three broad options when considering how to deal with JavaScript:

- **Don’t** Much of the AJAX content could be downloaded directly by your scraper. AJAX is generally sent as JSON, which means it is very easy to parse. You could save yourself a lot of time if you spent some time evaluating the target more closely.

- **Do it offline** Under this approach, you download the content, send it to a JavaScript interpreter such as SpiderMonkey, then process the results. If this sounds like a lot of manual work, it is. Fortunately for us, other people have struggled with this problem before and have released software to take care of most of the detail. Take a look at crowbar and webkitcrawler.

- **Automate a browser** This third approach involves relying on a web browser’s handling JavaScript itself. Until recently, this has involved quite a bit of complicated effort. Now, a library called splinter has come along to make life much easier.

One of the biggest differences between the second and third options is that the second option does not require a monitor. That means, it can be much easier to deploy on a server. However, in general the tasks we’ll be doing are fairly small and can happily run in the background while you’re doing other work.
Path of least resistance - splinter  Splinter is Python library that takes all of the trouble out of this process:

>>> from splinter.browser import Browser
>>> br = Browser('webdriver.chrome')

As a trivial example, let’s find Auckland’s current weather from the New Zealand Herald. If you visit their homepage without JavaScript enabled on your internet browser, you’ll see nothing. However, with JavaScript, an icon appears

>>> br.visit('http://www.nzherald.co.nz/')
>>> high = br.find_by_css('span.high').first.value
>>> low = br.find_by_css('span.low').first.value
>>> high, low
'19\xb0', '11\xb0' # \xb0 is the degree sign

Dealing with PDF content  PDF documents are a pain. Some PDF generators don’t actually have the concept of a word– every letter is individually placed. This makes it very hard to create a software tool that can combine letters to make words, and combine words to make sentences. However, depending on the source documents, there are possibilities for extracting information from them.

The ‘Data Science Toolkit’ is now the best way to get up and running with these kinds of tasks. Its “File to Text” tool takes an image, PDF or MS Word document and returns text to you.

If you only have a few documents to process, the website actually allows you to do the processing on their servers.

Extracting plain text  A quick way to extract text from a PDF programmatically is with the Python library, ‘slate‘. Disclaimer: I maintain slate. Its philosophy is to have a very low barrier to entry, but only extracts plain text out of the document:

>>> import slate
>>> with open('salesreport.pdf') as f:
...    report = slate.PDF(f)
...
>>> report[0]
"2011 ...
"

Digging deeper  One of the better free tools is called `pdftohtml`. It generates an HTML version of the document, which can then be processed by tools that you are used to. It does a good job of understanding the layout.

It is possible to circumvent security measures in PDF documents. The PDF viewer ‘xpdf’ provides this by default. This allows you to print or extract content that may be otherwise prevented through security measures.

General Tips

Avoiding being blocked  It’s possible to use sophisticated techniques to circumvent rate limitations and IP address blocking. However, the best technique for avoiding being blocked is by being a good netizen and adding pauses between your requests.

Scrape during the night of the site’s local time. This is very likely to have very few users, meaning the site will have more capacity to serve your scraper.

Be part of the open data community  When scraping open data, you should use ScraperWiki. ScraperWiki allows people to cooperatively build scrapers. They will also take care of rerunning your scraper periodically so that new data are added.
By being part of the community, you increase your profile, learn much more and benefit from people fixing your scraper when it breaks.

**Learn async programming**  Network programming is inherently wasteful in many ways. Your processor is consistently waiting for things to arrive from other parts of the world. Therefore, you can speed up the processing steps of your scrapers significantly if you take the time to learn asynchronous programming.

**Scraping multiple Pages using the Scraper Extension and Refine**

Many times you’ll not have one single page to scrape. For example the Chilean Government has a very nice transparency site and offers the income statistics for many departments - let’s get them all!

To complete this recipe you’ll need:
1. The Scraper Extension for Chrome
2. A Google Account
3. Refine
4. If you haven’t yet: Look at the Recipe Scaping websites using the Scraper Extension

To extract information out of multiple web-pages we’ll use a two step procedure: First we’ll get all the URLs for the web-pages with the scraper extension, then we will extract the Information out of these web-pages using Refine. To do this effectively, we rely on all the web-pages to be generated with similar structure. As you’ll see in this Recipe, it is not always the case.

**Walkthrough: Getting a list of URLs with scraper extension**

1. Open [http://www.gobiernotransparentechile.cl/directorio/entidad](http://www.gobiernotransparentechile.cl/directorio/entidad) - the list of government departments in Chile.
2. Mark one department, right click and select “scrape similar...” as in Scraping websites using the Scraper Extension
3. Depending on your selection you might have slightly different results.
4. Now pay attention to the left side of the scraper console - on the top you’ll see a text entry saying XPATH
5. The XPATH tells the computer where to find things. The text refers to html tags in the page - let’s modify this to make it work nicely for the page: First remove the `[/]` behind the second div. This will give you all the texts on the page.
6. Now let’s extract the links from the page. Go back to the page and “inspect” one of the links
7. See how the links are within an `<a>` element underneath the `<li>` element? Let’s add this to our xpath.
8. We have a column now with the names - if you look closely in the inspected element the `<a>` tag has two attributes: class and href. Href is the URL and class says something about the category the link belongs to. Let’s extract both.
9. To do so we’ll add more columns to the column list on the second page, do so by clicking the green `+` next to the one existing column
10. The scraper console wants two things: An Xpath and a name: enter `@href` to the xpath and URL as a name
11. Now let’s do the same for the class. Enter `@class` and Class
12. Perfect - click on Scrape (or press enter) to see how your list will look like. Nice isn’t it?
13. Now save it to google docs.

Once we have the list of URLs let’s go to Refine to scrape the salary pages we can find easily.
Walkthrough: Scraping multiple pages using Refine

1. Download the Spreadsheet as CSV from Google Spreadsheets
2. Open Refine - it will open a browser window at http://127.0.0.1:3333
3. Now select Create Project and this Computer
4. Click on Choose Files and select the file you just downloaded.
5. Click Next - this will open the Preview in Refine
6. Refine should parse the file correctly - name your project on the top right and click Create Project
7. If the special characters in the file look garbled - select UTF-8 as a Character Encoding.
8. You'll see the data view of Refine. It looks very much like a spreadsheet - and works quite similar.
9. Let's clean our data and only get the links we're interested in - the secondary category links
10. Create a filter for the Class column: click on the small triangle next to the column and select Facet -> Text Facet
11. This will open a Facet window in the left column that displays all the unique values found within the Class column, along with a count of how many times each of them appear in the column.
12. Let's remove everything but secondaryCat. Select secondaryCat in the filter and then click on the small invert label on the top right of the facet.
13. Now let's remove the rows that are not secondaryCat - for this select the options in the All column and select edit rows - remove all matching rows
14. Perfect - now we can delete the Class Column, since we're not going to use it anymore.
15. Do this with the Edit columns -> re-order/remove columns options in the All menu
16. Now let's scrape the pages. If you look at the page structure, the salary information is often in: /per_planta/Ao-2013 relative to the URL we scraped with the scraper extension.
17. Let's import the pages - do so by selecting the options of the URL column, and edit column -> add column by fetching URLs
18. This will open an add column menu. We have to transform our cells so that we get the url we want to be fetching.
19. Let’s write our first expression in Refine: we want to append /per_planta/Ao-2013 to each url, so our expression will be value+'/per_planta/Ao-2013' - this appends to the value in the current row.
20. The throttle delay sets the rate at which OpenRefine will request the pages from the webserver they live on. If we try to grab too many pages in a short period of time, the server may lock us out. Requesting 1 or 2 pages a second (throttle delay of 1000 or 500 milliseconds) should be okay in this case. Just be respectful:-). We also need to give the column we're creating for the scraped data a name - put Page into the column name.
21. click OK - refine will now download the webpages and give you a progress indicator, how it is doing. If you multiply the throttle delay by the number of rows in your dataset, you should get an estimate of the amount of time the download is likely to take.
22. Wooha, this quickly filled our document with HTML code - don’t be intimidated you don’t need to understand or read it - the computer will do this for you.
23. Now let’s first remove all the rows that didn’t get a page (because there is none...) (we could also go and investigate and see whether there’s information we missed - but for the sake of simplicity let’s just delete them).
24. Create a text facet on the Page column, select blank and remove all matching rows as we’ve done above.
25. Close the text facet again to continue working. Now let’s extract the table rows out of each page. Do so by selecting the options of the Page column and edit column -> add column based on this column
26. Name the new column Rows
27. The expression for this is slightly more complicated: first we tell refine that this is an HTML document we do so by starting with `value.parseHtml()`.

28. Then we append `.select("table")[0]` - this will select the first table from it - see how the content in the new column changes?

29. Now we select all the rows in this column with `.select('tr')` - remember `tr` is the tag for the rows...

30. This will result in a list of rows - however refine can’t really handle lists. So we have to `join` the list - so we append `.join("|")` to join the list with a pipe character – the vertical line. (I choose this character because it usually doesn’t appear in the text).

31. Your expression should now look like this:

32. It looks complicated but is actually very simple - it’s just a row of commands - similar to spreadsheet formulas. If you struggle understanding it: Try to read it from the beginning (go back a couple of steps and try to see how the commands are chained together).

33. click `OK` to fill the `Rows` column.

34. Now remove the blank rows again as we did before.

35. Let’s split the `Rows` into actual `Rows` in refine

36. Select `Edit Cells -> Split multi-valued cells` from the `Rows` menu

37. A menu will pop up asking us for what kind of character we want to split at: we want to split at `|`

38. Now let’s extract the data cells in each row. We do this by `edit column -> add column based on this column` like we did above.

39. Well do `value.parseHtml()` again - to tell refine this is HTML. Then we `.select("td")` to select the data cells and we’ll join them again with the `|` character. The expression should look like this:

40. Click `OK` - you’ll notice now we have quite a bit of blank cells - before we remove them we have to make sure we have the department name with each row. - for this switch back to the column saying `Text` - this is the department names.

41. Select `Edit cells -> fill down` from the options menu.

42. This will put the thing it finds in the row into each subsequent empty row until it does find a new filled row.

43. Now you can delete the blank rows.

44. So let’s go back and get the table data out. First we’ll strip the HTML tags from the text, since we don’t need them anymore.

45. We can do so by transforming the cell:

46. Select `edit cells -> transform` from the options of the `Data` column.

47. We want to replace anything between pointy brackets with nothing: the expression for this is `value.replace(/<.*?>/,,)`

48. This has removed the HTML tags. If you look at the data: there is a further problem. A lot of the text has things like &Ntilde; we can parse this automatically

49. Select `edit cells -> common transforms -> unescape HTML entities` from the options of the `Data` column - now let’s split the data in this column into multiple columns

50. For this choose `edit column -> split into multiple columns` and add the `|` character as the split character for the columns.

51. Now let’s look at what we’ve got: we have multiple columns with the data as presented in the table
52. One of the columns contains the amount - as you can see: it’s not always a number: The multiple . make refine think it’s not a number: let’s fix this: we’re transform the column

53. The expression for this is ‘value.replace(‘.’,”‘‘).toNumber()

54. Wonderful - There is still some things wrong: e.g. the special spanish characters don’t translate well - now you know how to use the replace function to replace them.

55. The last thing we do is remove the columns we don’t need anymore: URL, Page and Row

56. Congratulations you’ve scraped and cleaned up a large dataset from several pages on the web.

Extracting Data from PDFs using Tabula

PDFs can be all forms and shapes - if you’re facing a nicely formatted PDF that is not scanned give Tabula a shot to extract the information. How? read the short walkthrough below:

You’ll need:

- Tabula http://jazzido.github.io/tabula/

Walkthrough: Extracting data from PDF tables

2. Start Tabula (most likely by double clicking on the tabula icon)
3. point your browser to http://127.0.0.1:8080
4. Choose the file you want to upload and click Submit
5. Wait until the PDF is fully loaded
6. Scroll down to page 167 - we’ll extract that table.
7. Click and pull a selection box over the table
8. A window will pop up to show how Tabula would extract the data.
9. Now download the Data as CSV
10. Fantastic you liberated the table from the PDF. Quick and easy wasn’t it?

Cleaning up Data Scraped from the Web

Section author: Tim McNamara <tim.mcnamara@okfn.org>

This section deals with how you can clean up data - having extracted it from the web by scraping.

Useful clean up steps One advantage of scraping data from the web is that you can actually have a better dataset than the original. Because you need to take steps to understand the dataset’s inconsistencies, you can eliminate or at least minimise them. From another perspective, spending time cleaning up messy data can fill the large gaps that your processor will experience when waiting for it to be downloaded from its host.

This section provides an example of several useful clean-up operations.

- Cleaning HTML
- Strip whitespace
• Converting numbers to number types:
• Converting Boolean values: ‘Yes’ -> True
• Converting dates to machine-readable formats: “24 June 2004” -> “2004-06-24”

Clean the HTML  HTML you find on the web can be atrocious. Here’s a quick function that can help. We make use of the ‘lxml’ library. It’s very good at understanding broken HTML and will render a perfectly-formed page for your extractor functions.

You may be concerned that this is computationally wasteful. This is true, but it can reduce lots of the irritation of extracting specific information from messy HTML:

```python
def clean_page(html, pretty_print=False):
    ""
    >>> junk = "some random HTML<br> for you to try to parse</p>"
    >>> clean_page(junk)
    '<div><p>some random HTML</p><p> for you to try to parse</p></div>'
    >>> print(clean_page(junk, pretty_print=True))
    <div>
    <p>some random HTML</p>
    <p> for you to try to parse</p>
    </div>
    ""
    from lxml.html import fromstring
    from lxml.html import tostring
    return tostring(fromstring(html), pretty_print=pretty_print)
```

Converting yes/no to Boolean values  Computers are far better at interpreting Boolean values when they are consistently provided. Irrespective of the programming language, normalising these values will make any automatic comparisons much richer:

```python
def to_bool(yes_no, none_to_false=True):
    ""
    >>> to_bool("")
    False
    >>> to_bool(None):
    False
    >>> to_bool('y')
    True
    >>> to_bool('yip')
    True
    >>> to_bool('Yes')
    True
    >>> to_bool('nuh')
    False
    ""
    yes_no = yes_no.strip().lower()
    if not yes_no.strip() and none_to_false:
        return False
    if yes_no.startswith('y'):
        return True
    elif yes_no.startswith('n'):
        return False
```

Converting numbers to the correct type  If you’re extracting numbers from HTML tables, they will each be represented as a string or Unicode, even though it would be more sensible to treat as integers or floating point numbers:
def to_int(number, european=False):
    ""
    >>> to_int('32')
    32
    >>> to_int('3,998')
    3998
    >>> to_int('3.998', european=True)
    3998
    ""
    if european:
        number = number.replace('.', '')
    else:
        number = number.replace(',', '')
    return int(number)

def to_float(number, european=False):
    ""
    >>> to_float(u'42.1')
    42.1
    >>> to_float(u'32,1', european=True)
    32.1
    >>> to_float('3,132.87')
    3132.87
    >>> to_float('3.132,87')
    3132.87
    >>> to_float('(54.12)')
    -54.12
    Warning
    -------
    Incorrectly declaring 'european' leads to troublesome results:
    >>> to_float('54.2', european=True)
    542
    ""
    import string
    if european:
        table = string.maketrans(',.','.,')
        number = string.translate(number, table)
        number = number.replace('.', '')
        if number.startswith('(') and number.endswith(')'):
            number = '-' + number[1:-1]
        return float(number)

If you are dealing with numbers from another region consistently, it may be appropriate to call upon the locale module. You will then have the advantage of code written in C, rather than Python:

>>> import locale
>>> locale.setlocale(locale.LC_ALL, '')
>>> locale.atoi('1,000,000')
1000000

**Stripping whitespace** Removing whitespace from a string is built into many languages string. Removing left and right whitespace is highly recommended. Your database will be unable to sort data properly which have inconsistent treatment of whitespace:
Converting dates to a machine-readable format  

Python is well blessed with a mature date parser, dateutil. We can take advantage of this to make light work an otherwise error-prone task.

dateutil can be reluctant to raise exceptions to dates that it doesn’t understand. Therefore, it can be wise to store the original along with the parsed ISO formatted string. This can be used for manual checking if required later.

Example code:

```python
def date_to_iso(datestring):
    """
    Takes a string of a human-readable date and returns a machine-readable date string.
    """
    from dateutil import parser
    from datetime import datetime
    default = datetime(year=1, month=1, day=1)
    return str(parser.parse(datestring, default=default))
```

Sorting Data with Spreadsheets

This tutorial uses Google spreadsheets to sort data. Other spreadsheet programs work in a similar way - play around and see how they differ.

There is sample data for this tutorial here.

Walkthrough: Sorting a dataset.

1. Select the whole sheet you want to sort. Do this by clicking on the right upper grey field, between the row and column names.
2. Select “Sort Range...” from the “Data” menu – this will open an additional Selection.
3. Check the “Data has header row” checkbox.
4. Select the column you want to sort by in the dropdown menu.
5. Try to sort by GDP – Which country has the lowest?
6. Try again with different values, can you sort ascending and descending?

Tip: Be careful! A common mistake is to forget to select all the data. If you sort without selecting all the data, the rows will no longer match up.

Filtering Data

This tutorial uses Google spreadsheets to filter data. Other spreadsheet programs work in a similar way - play around and see how they differ.
There is sample data for this tutorial here.

Filtering data in a spreadsheet allows you to shut out the values you don’t want to see. For example, in the sample data, some “Country Names” are actually not countries? You’ll find things like “World”, “North America” and “Arab World”. Let’s filter them out.

**Walkthrough: Filtering Data**

1. Select the whole table.
2. Select “Filter” from the “Data” menu.
3. You now should see triangles next to the column names in the first row.
4. Click on the triangle next to country name.
5. You should see a long list of country names in the box.
6. Find those that are not a country and click on them (the green check mark will disappear).
7. Now you have successfully filtered your dataset.
8. Go ahead and play with it - the data will not be deleted, it’s just not displayed.

**Spreadsheet Formulae**

This tutorial uses Google spreadsheets to analyse data. Other spreadsheet programs work in a similar way - play around and see how they differ.


**A quick introduction to common spreadsheet symbols**

Now that you have a sense of how spreadsheet formula work, here’s a quick introduction to some of the most common formula symbols that you are likely to come across.

**The symbols**

These are all ‘basic maths functions’ - the kind of things you would find on a simple calculator.

- `=` Tells your spreadsheet that you are writing a formula. This is the first thing that should go in your formula cell.
  
  (NOTE: A spreadsheet assumes that *everything* that begins with an ‘=’ is a formula... so be careful how you use it!)

- `+` Add
- `-` Subtract
- `*` Multiply (this would be ‘x’ on a calculator)
- `/` Divide (this would be ‘÷’ on a calculator)

**Tip: Get your symbols in the right order**

It is worth remembering that basic maths rules about the order of functions apply. For example, the formula `=3+5*2` will give you 13, NOT 16. If you’re not sure why or can’t quite remember the rules, check out [this basic introduction](http://bit.ly/Y11xkF). If you want to change the order of function you’ll need parentheses: Formulas inside parentheses will be evaluated before any other formula. If you want the formula above to result in 16 you’ll need to type: `=(3+5)*2`

Have a go at using these formula in the ‘play sheet’ of your spreadsheet until you feel comfortable with them. You should find that they work pretty much as you would expect them to.
What if you wanted to add more numbers? You could always add them manually using + or you could use \textit{SUM} a formula to sum up all the values in the given range. Let’s try to calculate how many apples, plums and total fruit we sold during the week: Go to cell B7 and type \texttt{=SUM(A2:A6)} this will add the numbers of apples.

\textbf{Walkthrough: Using spreadsheets to add up values.} Let’s calculate the total of fruits sold.

1. Get the \texttt{example data} and create a copy.
2. To start, move to the first row.
3. Each formula in a spreadsheet starts with =
4. Enter = and select the first cell you want to add. Notice how the cell reference appears in the formula?
5. now type + and select the second cell you want to add
6. Press \texttt{Enter} or tab.
7. The formula disappears and is replaced by the value.
8. Try changing the number in one of the original cells (apples or plums) you should see the value in total update automatically.
9. You can type each formula individually, but it also possible to cut and paste or drag formulas across a range of cells.
10. Copy the formula you have just written (using \texttt{ctrl+c}) and paste it into the cell below (using \texttt{ctrl+v}), you will get the sum of the two numbers on the row below.
11. Alternatively click on the lower right corner of the cell (the blue square), and drag the formula down to the bottom of the column. Watch the ‘total’ column update. Feels like magic!

\textbf{Walkthrough: Using a spreadsheet to subtract values.} Take a look at the above example - it’s exactly the same, apart from you use a – instead of a +. Easy! Find the difference between the two columns in the example above.

\textbf{Walkthrough: Multiplication and Division}

\textbf{Math recap:} If you have a percentage and the value it is associated with you can calculate the value of the percentage. e.q. let’s say 25\% of people in a town of 1000 inhabitants are below 15 years. You can calculate the number of inhabitants by using the following formula:

\begin{align*}
\text{number of inhabitants in a town} & \times \frac{25}{100} \\
&= \frac{25 \times 1000}{100}
\end{align*}

(For more thorough explanation of percentages check out \texttt{BBC Skillswise})

\textbf{Now for a more complicated example:}

Let’s go back to our World Bank dataset. If you haven’t done the previous tutorials or lost the file: download it \url{here}.

1. In our original data, we have three columns related to health expenditure; ‘health expenditure (private)’, ‘health expenditure (public)’ and ‘health expenditure (total)’. So we’ll need to add three new columns to the right of the spreadsheet to do your calculations. Give them each a heading; perhaps ‘health expenditure (private) in USD’ etc.

2. In the original data, public, private and total healthcare expenditure is expressed as a \% of GDP. The GDP is already given in USD. To work out the expenditure in USD from these two facts is just one step. We want to calculate how much money (in USD) is spent on healthcare per country and year.
3. Let’s start by looking at the very first complete row (NB: spot the gap! we don’t have the data for Afghanistan’s GDP in 2000. Just be aware of this for now (we will talk in more detail about gaps in data later). The first complete row is Afghanistan in 2001.

4. In 2001, Afghanistan’s GDP was $2,461,666,315. Their private healthcare expenditure was 6.00937744% of this. So the calculation you need to do is

\[ 2461666315 \times 6.00937744 / 100 \]

1. With a spreadsheet formula, we don’t have to worry about all the numbers - you just need to enter the cells. So the formula you are going to need is: =E3*H3/100 (where cell E3 contains Afghanistan’s GDP in 2001, and cell H3 contains private health expenditure in Afghanistan in 2001).

2. Drag this formula all the way down the column and hey presto! You should have calculated the private health expenditure in USD for every country for the past 10 years. Much quicker than doing all the sums yourself!

### Walkthrough: Copying formulae sideways

In the same way as we could drag the formula down the column and the spreadsheet recognised the pattern and chose the correct cells, we can also drag the formula sideways to the new columns (public health expenditure in USD and total health expenditure in USD). So, if we want BUT we need to make one minor adjustment.

Still using the World Bank Data, try just dragging a cell formula across. Can you see the problem? The spreadsheet automatically moves all the cells its looking at one column to the right. So whereas before we had:

\[ =E3*H3/100 \]

we now have

\[ =F3*I3/100 \]

...but GDP is still in column E, so this formula is not the one we want.

To ‘fix’ a column or row, all you need to do is add ‘$’ in front of the section you want to fix. So, if you adapt your original formula to...

\[ =$E3*H3/100 \]

...you should be able to drag it over to the right without any problems.

**Tip:** It can be a little confusing getting used to the $ command at first. If this is the first time you’ve come across it, we suggest you spend some time playing around and seeing what it can do. Go back to your ‘play’ spreadsheet, make up some numbers, and experiment! Try for example =$B2*C2 vs =$B$2*C2, drag it around, and see what difference that makes. The best way to get comfortable with formulae is to use them!

So now, with one simple formula, you can calculate the actual expenditure of public, private and public+private healthcare, in every country, for the past ten years. Spreadsheets are pretty powerful things.

### Walkthrough: Minimum and Maximum Values

One thing that is very interesting to us when working with data is the maximum and minimum values of each of the columns we have. This will help us understand if the values are close together or far apart. Let’s do this!

1. Open a new sheet. Do so by clicking the “+” in the lower left corner

2. Leave the first column in the first row blank, in the second column enter = to tell the spreadsheet you will be using a formula.
3. Switch back to the sheet with your World Bank dataset.

4. Select the first column that has numerical data on the sheet where your data lives.

5. Press enter and you will see the name in the first sheet: magic. Why do we do it like this and not simply copy and paste? This will automatically change the headings if you change your headings (e.g., you move columns around or rename things).

6. Now the first column is going to be what you calculate: type Minimum in the second row first column (A2) for the minimum value.

7. In the cell right next to it type `=MIN(` (MIN is the formula for minimum)

8. Go back to the other sheet to select the first column with numerical data - to select the whole column click on the grey area with the column letter.

9. Close the brackets by typing `)`.

10. You should now see the minimum value in that field.

11. Now do the same for Maximum in the third row. Once you are done, just mark the three values in the second row (the formula for maximum is `=MAX()`).

12. See the blue square in the right lower corner? Grab it and pull it right. Release it and if you still not have all columns, carry on until you have all values.

13. This way you created a table with the minima and maxima of each of the columns.

Walkthrough: Dealing with empty cells  Did you notice some of the minimum values are 0? Do you really believe there are countries not spending money on healthcare? There aren’t (well, probably). The zeroes are because there are empty cells. Properly handling missing values is an important step in data cleaning and analysis - hardly ever are large datasets complete and you have to find a strategy to deal with missing parts.

In this walkthrough we will create a complex formula. We will do so with an iterative process - this means one little formula at the time. If you follow us through you’ll notice you can create quite complex formulas and results simply step by step.

1. To deal with empty cells we have to fix parts of our calculation formulas in the World Bank datasheet.

2. To start - create a mock spreadsheet to play with data. Copy the first few rows of the World Bank dataset into it so you’ll have a start. To validate our formulas try to remove values in some of the rows.

3. We got a missing problem right in the first value: Afghanistan’s GDP is missing for the year 2000.

4. Think about our goal. What we want to achieve: if either of the values we are multiplying (in this case, GDP or health expenditure) is not a number (probably because the value is missing), we don’t want to display the total.

5. To put it another way: only if a value for both GDP and healthcare expenditure is present should the spreadsheet carry out the calculation; otherwise it should leave the cell blank.

6. The formula to express this condition is ‘IF’. (You can find an overview on formulas like this on the google doc help.)

7. The formula asks us to fill out the three things: (1) Condition, (2) value if the condition is true, (3) value if the condition is false.

   `=IF(Condition, Value if condition is true, Value if condition is false)`
8. In our case we know parts (2) and (3). (2) is the formula we used above this is the calculation we want to carry out if both values are present in the spreadsheet.

\[=\text{IF(Condition, } \$E3\times H3/100, \text{ Value if condition is false)}\]

9. (3) is a blank - if the numbers aren’t there, we don’t want to display anything, so we fill in that value with nothing at all.

\[=\text{IF(Condition, } \$E3\times H3/100,)\]

10. So now we just need to work out (1), the condition.

\[=\text{IF(Condition, } \$E3\times G3/100,)\]

11. Remember that we want the condition to be that BOTH the GDP and healthcare expenditure values are a number. The formula to see whether a cell is a number is: ISNUMBER.

12. This is another one of those little formulas that you should try playing with! If you type =ISNUMBER(F2) and F2 is an empty field, it will say FALSE. If there is a number it will say TRUE. Handy isn’t it?

13. We want a formula that will only be calculated if both GDP and healthcare expenditure are actual numbers.

14. We need to combine the results of both ISNUMBER(GDP) and ISNUMBER(healthcare expenditure) together. The formula to do so is AND. This will simply say TRUE if both of them are TRUE (i.e. both of them numbers) or FALSE if either one or both of them is FALSE.

15. Which is exactly what we need. So our condition will be:

\[\text{AND(ISNUMBER(gdp),ISNUMBER(healthcare expenditure))}\]

16. or, to use our cells from before

\[\text{AND(ISNUMBER($E3),ISNUMBER(H3))}\]

17. Phew! So now we can put parts (1), (2) and (3) from above all together in one big formula, using ‘IF’

\[=\text{IF(Condition, } \$E2\times H2/100,)\]

\[=\text{IF(AND(ISNUMBER($E2),ISNUMBER(H2)),} \$E2\times H2/100,)\]

18. Try it out: enter it to the first row of the first column of the calculation and paste it to all the other places. It should leave the cells empty.

If you look at the data you will quickly find out that countries with higher number of people spend more on healthcare than countries with lower number of people. Intuitive isn’t it. So how to compare the countries more directly? Break it down to healthcare expenditure per person! This step is called normalization and is a step often done when comparing different entities - such as countries etc.

**Geocoding Data in a Google Docs Spreadsheet**

A very common need is to geocode data in a Google Spreadsheet (for example, in creating TimeMaps with the Timeliner project). There are several options here:

By hand – use a Geocoding service (see the course on geocoding) and then copy and paste by hand. Use the ImportXML (or ImportCSV) formulae to grab data from a geocoding service – great but with limitations on the number of rows you can code at one time (~50). Use a Google App Script – the most powerful but requires installation of an App Script in your spreadsheet. In this tutorial I’m going to cover the latter two automated options and specifically focus on option 2.

Using Formulas All of the following is illustrated live in this google spreadsheet.
We start with a formula like the following:

```
```

This formula uses the ImportXML function to look up XML data from the Mapquest Nominatim geocoding service (see the previous tutorial for more about geocoding services). The first argument to ImportXML is the URL to fetch (in this case the results from querying the geocoding service) and the second part is an XPath expression to select data from that returned XML. In this case, the XPath looks up the first place object in the results: place[1] and then gets the lat (latitude) attribute. To understand this more clearly, here’s the XML returned by that XML query:

```
<place>
  <lat>51.5074</lat>
  <lon>-0.1278</lon>
</place>
```

In reality we want both latitude and longitude, so let’s change it to:

```
```

This uses an “or” || expression in XPath and the result will now be an array of results that Google Docs will put in 2 cells (one below another). You can see this in Column C of the example spreadsheet.

What happens if we wanted the data in just one cell, with the two values separated by commas, for example? We could use the JOIN function:

```
```

Lastly, we’d like to geocode based on a place name in an another cell in the spreadsheet. To do this we just need to add the place name to our API request to MapQuest’s Nominatim service using the CONCATENATE function (this example assures the value is in cell A2):

```
=ImportXML(CONCATENATE("http://open.mapquestapi.com/nominatim/v1/search?format=xml&q=", A2), "//place[1]/@lat")
```

```
=JOIN("", ImportXML(CONCATENATE("http://open.mapquestapi.com/nominatim/v1/search?format=xml&q=", A2), "//place[1]/@lat")
```

**App Script**  
If you want an even more powerful approach you can use a Google App Script. In particular, Development Seed’s MapBox team have prepared a great ready-made Google AppScript that will do geocoding for you.

Find the *script plus instructions* online.

**Using a spreadsheet to clean up a dataset**

This recipe was created for the School of Data by Tactical Technology Collective. Tactical Tech is an international NGO working at the point where rights advocacy meets information and technology.

**Figure 1.3:** Cleaning. Sometimes challenging. Even with the right tools. Image: *Bath Time* by archer10. Some rights reserved CC-A-SA 2.0.

**Introduction**

**What is this recipe and what do you get out of it?**  
Cleaning data is an essential step in increasing the quality of data. This Data Wrangling Handbook recipe looks at six common ways that a dataset is ‘dirty’ and walks through time-saving ways you can use a spreadsheet to fix them and ‘clean’ the dataset.

The recipe is very detailed, because data cleaning is all about attention to detail. Some of it is easy - and you’ll wonder why we included it! But some of it is hard - particularly the later sections on structural problems and inconsistencies. But by the end of it you will have used a set of spreadsheet features and functions in combination with each other to do something useful.
It will help data cleaning become **less tedious** by showing you how to use the software to do thing you’d otherwise do ‘manually’, with the accompanying risks of missing some errors, and introducing new ones. The techniques we introduce also **scale** to a certain point. The sample dataset we use has 400 or so rows of data, which is just big enough to be a headache to clean up by going row by row, fixing problems by hand. These techniques will work where you have a lot more data.

It is aimed at people who are familiar with spreadsheets, and requires no knowledge of programming. It uses a range of features and functions of the spreadsheet. The spreadsheet features we will use include:

1. Autofilter - quickly find out what’s in a column of data, and show which bits you want
2. Conditional formatting - highlight cells of data based on criteria you specify
3. Data Validation - choose what values are allowed in a cell
4. Defined data ranges - create lists of data that can be used to make comparisons
5. Find and Replace - look for one thing, and replace it with another
6. Paste Special - remove unwanted formatting
7. Pivot tables - summarise your data and see it in completely new ways
8. Regular Expressions - clever way of searching for data
9. Text-to-Columns - split up cells of data that have more than one entry in them

The recipe also makes use of forumulae that use some different spreadsheet functions, including CLEAN, CONCATENATE, COUNTA, IFBLANK, LEFT, SEARCH, SUBSTITUTE, and TRIM. You may not have known these existed, but they can speed up a lot of the most difficult and time-consuming parts of removing errors from a dataset.

The recipe is a complete process, showing every step. It can be a useful learning aid, guide or reference point for you in managing your own datasets. Finally, this recipe is the ‘textbook’ for a School of Data Course called A Gentle Introduction to Data Cleaning which is a more accessible and engaging way into this quite dry topic.

**Get set up: data and tools you will need to follow this recipe**

**Software tools used in the recipe**  The spreadsheet software we have used throughout this recipe is part of Libre Office 3, which is free, open source software that works on Windows, Mac and Linux operating systems. Libre Office 3 can be downloaded and installed from the Document Foundation website. For help getting set up have a read of the free installation guides (they’re quite good).

Most of this recipe can be also be used in recent versions of Microsoft Excel and Google Spreadsheet. The screenshots will look different and the steps may not be quite the same though. To help complete some tasks, we will also occasionally dip into a word processor and text editor like Notepad. These will probably already be installed on your computer.

**The ‘dirty’ data - what we’re starting with**  We will be working with a spreadsheet of data about sales of vast amounts of agricultural land in less developed countries. The data was researched by a non-governmental organisation called GRAIN, which works to support small farmers and social movements in their struggles for community-controlled and biodiversity-based food systems.

We have placed a spreadsheet with the original dataset on the Data Hub, an online library for data. To follow this recipe, you will need to download a copy of this onto your computer.

**The ‘clean’ data - the end result**  By the end of this recipe, the data will be substantially cleaned up. We’ve put a copy of the original data along with the final, cleaned GRAIN data online too.
Problem 1: Showing the data plainly

What's the problem? Spreadsheets are visual tools that give you control over the way text and numbers are displayed: this is called formatting. It can be helpful, and help display the data in ways that make it easier to use and understand. It can also be unhelpful make data more difficult to see.

Before moving forward, look at how the GRAIN data is currently displayed:

- The text is very small. This is maybe to try and fit all the data on the screen so you don’t have to scroll right to see it.
- There is some colouring in there. Why? What does it mean? Does it signify something important about the data?
- Can you find any hidden rows or columns? Look for discontinuities in the column lettering and row numbering.

What's the solution?

Use the cell formatting features To display the data in a plain and compact way, removing any choices that the original publisher of the spreadsheet made, you can do the following:

1. Select the complete dataset (Linux/Windows shortcut: Ctrl+A).
2. With the data now selected, right click on top of any row label (eg. 1, 2, 3 etc) to bring up the secondary menu. Choose “Show”. This will make sure any rows that are hidden are revealed. Repeat the same steps for the columns.
3. With all the data selected, right click anywhere to show the secondary menu. Select “Clear Direct Formatting” to remove cell colouring and make the text size uniform.
4. Right click again, and go to “Format Cells”. Select the ‘Alignment’ tab and choose the following options to place the text sensibly in the cells:
5. With all the data still selected, right click on any row label again and choose “Optimal Row Height”, and select ‘Okay’. This will resize the rows to remove extra vertical space, which results in more data being displayed in-screen.
6. To resize the columns, place the mouse cursor in the line between columns. Left click and hold, and drag the line until you’re happy with the column width.

Problem 2: Whitespace and new lines - data that shouldn’t be there

What’s the problem? Apply autofilter to the GRAIN data in the worksheet (Data →Filter → Auto-Filter). Bring up the select list for “Status of Deal” by choosing the downwards-pointing triangle that has appeared in the column label, as below:

Why are there three entries for “Done”? Look at the selection list for other columns? Can you see where there are other duplicate entries?

The reason for the duplicate entries is that there are extra characters alongside the data that are not displayed - so you can’t easily see them. These are likely to be either whitespace at the ends of lines (also called ‘trailing spaces’ or new lines) that were added accidentally during the data entry. These are very common errors, and their presence can affect eventual analysis of the data, as the spreadsheet treats them as different entries. For example, if we are counting deals that are categorised as Done, the spreadsheet will exclude those that are categorised as “Done “ (note the extra space at the end).
Similarly, cells can also have **new lines** in them, the presence of which is obscured by the layout. For example, find the cell containing the text ‘Libyan Investors’. On a first glance it looks fine, but double click to edit it. There is an extra line beneath the word ‘investor’, as illustrated by the presence of the cursor beneath the text:

Entries for ‘Libyan investors’ with and without a new line afterwards are treated by the spreadsheet as different data. In turn, this will affect the accuracy of data analysis.

**What's the solution?** There are two easy ways to remove **whitespace** and **new lines** from a worksheet. Both are equally as effective.

**Use the Find and Replace feature** Both whitespace and new lines can be “seen” by the spreadsheet.

1. Open the find/replace tool (Shortcut: Ctrl-H).
2. Select “More Options” and check “Regular expressions”. This feature enables the spreadsheet to search for patterns, and not just specific characters.
3. In the input area for Find type `[:space:]$` and click “Find”. This is a regular expression that searches only for spaces that are at the end of the text in a cell (which is what the $ denotes). It should look like this:
4. Running this search will show you the cells in this worksheet that have one or more trailing whitespaces.
5. To remove the trailing whitespace that have been found, click “Find All”. Make sure the input area for ‘Replace with’ is empty. Then click on “Replace All”. Perform this operation until the spreadsheet tells you, “The search key was not found”.
6. To remove the new lines, repeat steps (a) to (e) with n in the Findbox. Remember to keep regular expressions enabled, or this won’t work.
7. Run the auto-filter again to see how the data has changed.

**Use the TRIM and CLEAN functions** Trailing whitespace and new lines are common enough problems in spreadsheets that there are two specialised functions - clean and trim - that can be used to remove them. This is a little more detailed, so follow the steps carefully:

1. In your spreadsheet, the GRAIN dataset should be in ‘Sheet1’. Create a new worksheet for your spreadsheet, called Sheet2.
2. In cell A1 of the new worksheet you have just created enter the following formula: 
   
   =CLEAN(TRIM(Sheet1.A1)) and press enter. This will take the content of cell A1 from Sheet1, which is your original data, and reproduce it in Sheet2 without any invisible character, new lines or trailing whitespace.
3. Find out the full data range of Sheet1: It will be A1 to I417. In Sheet2, select cell A1 and then copy it (Shortcut: Ctrl+C). In the same sheet select the range A1 to I417 and paste the formula into it (Shortcut: Ctrl+V). The complete dataset from Sheet1 will be reproduced in Sheet2, without any the problematic invisible characters.
4. To work further on this data, you will have to now remove the formulas you used to clean it. This can be done with the Paste Special operation. In Sheet2, select the complete dataset and copy it. Put the cursor in Cell A1, and then go to Edit → Paste Special. This enables you to choose what attributes of the cell you want to paste: we want to paste everything except the formulas, which is done by checking the appropriate boxes, as below, and then clicking Okay:
5. Double click on any cell, and you will see that it just contains data, not a formula. If you like, run through the steps outlined in Problem 1 to make the text ‘wrap’ in cells, and adjust the column widths.

**Problem 3: Blank cells - missing data that should be there**
What’s the problem? In many spreadsheets you come across there will be empty (“blank”) cells. They may have been left blank intentionally, or in error. Either way, they are missing data, and it’s useful to be able to find, quantify, display and correct them if needed.

What’s the solution?

Use the COUNTBLANK function This will enable you to show the number of blank cells, which helps you figure out the size of the potential problem:

1. Scroll to the end of the dataset. In a row beneath the data, enter the following formula: =COUNTBLANK(A1:A417) and press enter. This will count the number of blank cells in column A so far as row 417, the last entry in the GRAIN dataset.
2. In the same row copy the formula you just created across rows B to I. This will show you a count of the blank entries in the other columns.
3. You can see that by far the most blank cells are in column G, ‘Projected Investment’.

Use the ISBLANK function with the Conditional Formatting feature Blank cells can also be highlighted using conditional formatting and the ISBLANK function, changing the background colour of blank cells, so you can see where they are:

1. Select the dataset (cells A1 to I417), and open the ‘Conditional Formatting’ menu (Format → Conditional Formatting → Conditional Formatting). This spreadsheet feature allows you to automatically change the formatting (eg. font size, cell style, background colours etc) depending on the criteria you specify.
2. In the conditional formatting window, make the conditions look like the image below. To make the blank cells stand out more clearly, use an existing style or set up a new one by clicking the ‘New Style’ option, which takes you to a window where you can choose font type, size, background colour and so on.
3. Check what has happened. When the conditional formatting is applied the blank cells will be highlighted in red. Here’s how it looks when zoomed out a bit (View → Zoom, then enter 75% into the ‘Variable’ option):
4. To remove the conditional formatting: repeat steps (a) to (b) above, but define a clear style (or use the ‘default’) instead. Otherwise select the cell or range of cells and select “Clear Direct Formatting”.

Fill in empty values with the Find and Replace feature In the GRAIN dataset we do not know whether blank cells signify data that has been deliberately or accidentally left out. You may want to actively specify that the data is missing, rather than leaving a blank cell.

1. Select the complete data range (A1 to I417).
2. Open the find/replace dialogue (Shortcut: Ctrl-H). If you have already used this earlier, you will need to disable searching with regular expressions, because this causes the search to work differently. Do this by clicking ‘More Options’ and uncheck regular expressions.
3. Leave the “Search for” input box empty. Type “Missing” into the replace box (without quote marks). Click on ‘replace all’.
4. Every blank cell will now have the value “Missing” recorded in it. You can verify this using the COUNTBLANK function we outlined above.

Filling blank cells isn’t always useful and it’s important to choose the right term to denote a missing value. For example, in the context of ‘Proposed investment’ using the term ‘none’ to signify missing data is confusing. Readers may think this means you know there is no investment, rather than that there is no data.

Problem 4: Fixing numbers that aren’t numbers
Data Wrangling Handbook, Release 0.1

What’s the problem?  The GRAIN dataset has a column called Proposed Investment. This records the amount of cash paid for land in US Dollars. However they are recorded as text, not as numbers. This means the spreadsheet can’t use these values to do the mathematical operations required to make totals, averages, or sort the numbers from highest to lowest. Further, the data have not been entered in a consistent form. Here are some examples from the dataset:

- US$77 million
- US$30-35 million
- US$1,500 million
- US$ 2 billion
- US$57,600 (US$0.80/ha)

So the problem is twofold: there is no consistent unit, and there are data other than the currency in the cell. Ideally, what we would have are data like this:

<table>
<thead>
<tr>
<th>Projected investment (US$ millions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>77</td>
</tr>
<tr>
<td>32.5</td>
</tr>
<tr>
<td>1500</td>
</tr>
<tr>
<td>2000</td>
</tr>
<tr>
<td>0.057600</td>
</tr>
</tbody>
</table>

What’s the solution?  We can solve this with a combination of automation and old-fashioned hand correction of data. A part solution using a combination of formulas

2. Create a new column H called “Projected Investment (US$ millions) to the immediate right of the current column G, Projected Investment. We will use column H as a working column to display the outcomes of our calculations.
3. Go to Cell H2, and enter the following formula: =LEFT(SUBSTITUTE(G2,”US$”,””), SEARCH(” ”, SUBSTITUTE(G2,”US$”,””))) Then copy it (Shortcut: Ctrl-C).
4. Select the range H1 to H417 using the mouse, and paste the formula into that range. You will see that if there is any data in Column G, a new value will be displayed in column H, as below:

Where there is no data, a warning sign like #VALUE! will be shown.
5. This formula works using three functions joined together: LEFT, SUBSTITUTE and SEARCH.

That’s some crazy stuff, dude! Explain yourself.  It’s complex but a good exercise in thinking about what data is, and how spreadsheets can process text quite effectively by combining different functions into formulas.

Let’s start with the simplest and most common sort of case from the GRAIN database:

US$77 million

We want to turn this into a number that the spreadsheet can work with:

US$77,000,000

There are two things that we can immediately do: specify the currency as an attribute of all numbers in the column “Projected investment” so we know that all numbers in this column are US$. This removes the need to put the text “US$” in each cell:

77,000,000
As nearly all the entries are over 1 million in size, it’s sensible to specify that all numbers in the column “Projected investment” are in millions. This removes the need to include the trailing zeros - the 000,000 - in the cells. This leaves us with:

77

So, the actual task the formula needs to do is to change “US$77 million” to “77”. We want to remove everything but the number 77, with as little potential for error as possible and in a way that can be applied to as many of the other data in the ‘Projected Investment’ column as possible.

This is where the LEFT function comes in. Look at the value we want to change: US$77 million. Count the characters, including the spaces: there are 13 in total. The LEFT function reads the value, and displays only characters to the left of and including the character number you give it. Here’s how it works on the value “US$77 Million”:

<table>
<thead>
<tr>
<th>Formula</th>
<th>You see</th>
<th>Which is...</th>
</tr>
</thead>
<tbody>
<tr>
<td>=LEFT(&quot;US$77 million&quot;,2)</td>
<td>US</td>
<td>The first 2 characters</td>
</tr>
<tr>
<td>=LEFT(&quot;US$77 million&quot;,3)</td>
<td>US$</td>
<td>The first 3 characters</td>
</tr>
<tr>
<td>=LEFT(&quot;US$77 million&quot;,5)</td>
<td>US$77</td>
<td>The first 5 characters</td>
</tr>
</tbody>
</table>

In the above examples we included in the formula the actual text that we wanted to analyse using LEFT. We can specify which cell we want to analyse (this is called cell referencing). For example, in the spreadsheet we might have:

<table>
<thead>
<tr>
<th>row</th>
<th>G</th>
<th>Formula in column I</th>
<th>Outcome in column I</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>US$77 million</td>
<td>=LEFT(G22,5)</td>
<td>US$77</td>
</tr>
<tr>
<td>23</td>
<td>US$56 million</td>
<td>=LEFT(G23,5)</td>
<td>US$56</td>
</tr>
<tr>
<td>24</td>
<td>US$45 million</td>
<td>=LEFT(G24,5)</td>
<td>US$45</td>
</tr>
</tbody>
</table>

Building the formula this way enables it to be copied down a column, as the cell numbers will update automatically as the position of the formula changes. We can further improve the formula and remove some of the text that we ask LEFT to analyse. This is where the SUBSTITUTE function is useful. Here’s how it works, then we’ll apply it in combination with the LEFT function:

<table>
<thead>
<tr>
<th>row</th>
<th>G</th>
<th>Formula in column I</th>
<th>Outcome in column I</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>US$77 million</td>
<td>=SUBSTITUTE(G22,&quot;US$&quot;,&quot;&quot;)</td>
<td>77 million</td>
</tr>
<tr>
<td>23</td>
<td>US$56 million</td>
<td>=SUBSTITUTE(G23,&quot;US$&quot;,&quot;&quot;)</td>
<td>56 million</td>
</tr>
<tr>
<td>24</td>
<td>US$45 million</td>
<td>=SUBSTITUTE(G24,&quot;US$&quot;,&quot;&quot;)</td>
<td>45 million</td>
</tr>
</tbody>
</table>

The SUBSTITUTE function takes the content of a cell (eg. G22, which has the text US$77 million), looks in it for the specific text you tell it to (in this case “US$”), then substitutes it with what you tell it to (in this case, for ““, which is nothing at all) and prints the result (77 million).

We can combine SUBSTITUTE with LEFT. So, in the below, LEFT does its work on text that has already had characters removed through the SUBSTITUTE function:

<table>
<thead>
<tr>
<th>row</th>
<th>G</th>
<th>Formula in column I</th>
<th>Outcome in column I</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>US$77 million</td>
<td>=LEFT(SUBSTITUTE(G22,&quot;US$&quot;,&quot;&quot;),2))</td>
<td>77</td>
</tr>
<tr>
<td>23</td>
<td>US$56 million</td>
<td>=LEFT(SUBSTITUTE(G23,&quot;US$&quot;,&quot;&quot;),2))</td>
<td>56</td>
</tr>
<tr>
<td>24</td>
<td>US$45 million</td>
<td>=LEFT(SUBSTITUTE(G24,&quot;US$&quot;,&quot;&quot;),2))</td>
<td>45</td>
</tr>
</tbody>
</table>

So, we have the numbers we need now but there is a problem. Not all the original numbers recorded in ‘Projected Investment’ are 2 digits long. Here’s what happens if we run this formula on a more varied set of data in the G column:

<table>
<thead>
<tr>
<th>row</th>
<th>G</th>
<th>Formula in column I</th>
<th>Outcome in column I</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>US$7710 million</td>
<td>=LEFT(SUBSTITUTE(G22,&quot;US$&quot;,&quot;&quot;),2))</td>
<td>77</td>
</tr>
<tr>
<td>23</td>
<td>US$5.34 million</td>
<td>=LEFT(SUBSTITUTE(G23,&quot;US$&quot;,&quot;&quot;),2))</td>
<td>56</td>
</tr>
<tr>
<td>24</td>
<td>US$450 million</td>
<td>=LEFT(SUBSTITUTE(G24,&quot;US$&quot;,&quot;&quot;),2))</td>
<td>45</td>
</tr>
</tbody>
</table>

Uh oh! You can clearly see there are mistakes in the outcome column. This is because we have told LEFT to show only 2 characters each time (remember we have removed the “US$” using SUBSTITUTE, so LEFT doesn’t count
those). However, to show the correct figure for “US$7710 million” in row 22, LEFT would have to count 4 characters. So how can we give LEFT the correct number of characters?

Look at the values again. They have something else in common: yes, they have a space separating the number from the text “millions”. Its position will vary each time but we can find it tell the LEFT function to show it where the number ends in each case. The SEARCH function can be used to do this. It works by looking through data you give it for a character you specify, and then tells you the position of that character:

<table>
<thead>
<tr>
<th>row</th>
<th>G</th>
<th>Formula in column I</th>
<th>Outcome in column I</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>US$7710 million</td>
<td>=SEARCH(&quot; &quot;,G22)</td>
<td>8</td>
</tr>
<tr>
<td>23</td>
<td>US$5.34 million</td>
<td>=SEARCH(&quot; &quot;,G23)</td>
<td>8</td>
</tr>
<tr>
<td>24</td>
<td>US$450 million</td>
<td>=SEARCH(&quot; &quot;,G24)</td>
<td>7</td>
</tr>
</tbody>
</table>

So, in row 22, we are looking for a space (“ ”) in the text in cell G22 (US$7710 million). Count from the left, that space is in position number 8. We can give this number 8 to the LEFT function:

<table>
<thead>
<tr>
<th>row</th>
<th>G</th>
<th>Formula in column I</th>
<th>Outcome in column I</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>US$7710 million</td>
<td>=LEFT(G22,(SEARCH(&quot; &quot;,G22)))</td>
<td>US$7710</td>
</tr>
<tr>
<td>23</td>
<td>US$5.34 million</td>
<td>=LEFT(G23,(SEARCH(&quot; &quot;,G23)))</td>
<td>US$5.34</td>
</tr>
<tr>
<td>24</td>
<td>US$450 million</td>
<td>=LEFT(G24,(SEARCH(&quot; &quot;,G24)))</td>
<td>US$450</td>
</tr>
</tbody>
</table>

Note that these outcomes also include the space after the number. Let’s add the SUBSTITUTE function back into the formula: Wherever there is a cell reference (eg. G22) we can put a SUBSTITUTE function removing the text US$:

<table>
<thead>
<tr>
<th>row</th>
<th>G</th>
<th>Formula in column I</th>
<th>Outcome in column I</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>US$7710 million</td>
<td>=LEFT(SUBSTITUTE(G22,&quot;US$&quot;,&quot;&quot;), (SEARCH(&quot; &quot;,SUBSTITUTE(G22,&quot;US$&quot;,&quot;&quot;))))</td>
<td>7710</td>
</tr>
<tr>
<td>23</td>
<td>US$5.34 million</td>
<td>=LEFT(SUBSTITUTE(G23,&quot;US$&quot;,&quot;&quot;), (SEARCH(&quot; &quot;,SUBSTITUTE(G23,&quot;US$&quot;,&quot;&quot;))))</td>
<td>5.34</td>
</tr>
<tr>
<td>24</td>
<td>US$450 million</td>
<td>=LEFT(SUBSTITUTE(G24,&quot;US$&quot;,&quot;&quot;), (SEARCH(&quot; &quot;,SUBSTITUTE(G24,&quot;US$&quot;,&quot;&quot;))))</td>
<td>450</td>
</tr>
</tbody>
</table>

That explanation help?

**Refine the solution to remove the errors** Throughout this example, you can see how a useful formula can be built up to help solve the problems we outlined at the beginning. However, taking this approach leaves us with loose ends, for example:

- Where there is no data in column G, this formula will not know what to do, and will return a #VALUE! error, which makes it more difficult to use the new data in other calculations.
- If the value is US$22 billion rather than $22 million the formula will still return 22. The correct value in a column of data in US$ millions should be 2200.
- If the value is US$30-35 million, the formula will return the range 30-35, rather than a single value.
- Where a value is below a million, and has some additional explanation, such as in “US$57,600 (US$0.80/ha)”, the formula will return 57,600. In a column denoting US$ millions this would be a huge amount.

We use formulas to automate the process of cleaning data to the greatest extent possible but there will always be exceptions. The key is to know where the exceptions are and decide whether it is worth continuing to try and accommodate them with a formula, or whether to just correct them by hand. How do we do this? We can use a feature of the spreadsheet called a Pivot Table. This will help us find the troublemakers, how many of them there are, and whether we should continue to fix with formulas by hand.

**Use a Pivot Table to find errors and Autofilter to help fix them**
1. At this point your dataset should have the original Projected Investment data in column G, and the cleaned data in column H, which we named Projected Investment (US$ Millions).

2. Select column G. Go to **Data → Pivot Table → Create New**. In the window that appears, checked “Current Selection” and click on **OK**. A strange new window called **Pivot Table** will appear, which looks like the image below:

3. We can use this to find a list of the unique values in column G (Projected Investment), which will help us identify trouble-causing entries. Select the little grey rectangle near the top centre labelled “Projected...” and drag it to the white area called **Row Fields**. Select it again and drag it to **Data fields** too. You should see this:

   Notice that in the Data Fields, the little grey rectangle is now labelled “Sum - Projected Investments...”. We want to change this to “Count - Projected Investment ...”, so it shows us how many times each of the different values occurs in the dataset. To do this, double click on it. This window will appear:

   **Select Count.** Then click **Okay** in the pivot table window. A new worksheet will appear, containing a list of unique values from column G, along with the number of times each occurs.

4. This view of the data enables us to quickly scan down the list and see the problems. The count lets us know how much work it is likely to be to fix them. So, with a quick bit of analysis of this pivot table, we can see that of a total 416 rows of data in the GRAIN dataset only 106 values (that is 416 minus the 310 where data are not present) are recorded in the column for Projected investment. Of these 106, only 14 are NOT uniform like “US$34 million” or “US$1,876 million”. Here are the offending entries, which we’ve pulled out into a table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>US$1.2/ha/yr (after first 7 years) in Gambela and US$8/ha/yr (after first 6 years) in Bako</td>
<td>1</td>
</tr>
<tr>
<td>US$1.3 billion</td>
<td>1</td>
</tr>
<tr>
<td>US$1.6 billion</td>
<td>1</td>
</tr>
<tr>
<td>US$125,000/yr (land lease)</td>
<td>1</td>
</tr>
<tr>
<td>US$2 billion</td>
<td>2</td>
</tr>
<tr>
<td>US$205 million (half of fund)</td>
<td>2</td>
</tr>
<tr>
<td>US$3.1 billion</td>
<td>1</td>
</tr>
<tr>
<td>US$4 million (lease cost for 25,000 ha)</td>
<td>1</td>
</tr>
<tr>
<td>US$4/ha/yr (lease)</td>
<td>2</td>
</tr>
<tr>
<td>US$57,600 (US$0.80/ha)</td>
<td>1</td>
</tr>
<tr>
<td>US$8/ha/yr (lease)</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>14</td>
</tr>
</tbody>
</table>

1. We can fix these in about 5 minutes simply by identifying them in the original data, and changing them by hand so our formula can then process it. Head back to the worksheet you have your clean data in. A quick way to highlight these entries is to use the autofilter selection list that we used above.

2. Go through the list and make sure that only the exceptions we have identified are selected. Then click **OK**. This will change what the spreadsheet shows: only those rows that have in column G the data you’ve selected will be shown. You can now go through them one-by-one, changing the data in column G so the formula we made can work on it. You will see the results in Column H, which will update automatically. For example:

   US$30-35 mil → hand correct into an average: US$32.5 million → Formula returns 32.5

   US$2 billion → hand correct in US$2,000 million → Formula returns 2,000

   ...and so on.

   **Tip:** as you are updating original data, you may wish to keep a note of what you changed. Either create a column called “Notes”, and record the data there. Or, duplicate column G and name this new column “Projected Investment (un-altered)”. Or, where appropriate update the column called “Summary” with the data, which is the approach we have taken.

3. There are a few final steps to take to make the numbers in column H usable. Currently, our data in column G is a calculated value, not a number: in spreadsheet language, this means we can’t add them up yet! We need to
replace the formula with its result. This is easily done with the Paste Special feature we noted above.

• Select the whole of column H (or just H1 to H417 if you prefer).
• Copy it (Shortcut: Ctrl-C). Put the cursor at the top of column H, selecting Cell H1. Go to Edit → Paste Special. A window will appear, like this:
• We can choose what aspects of the selected data are pasted by checking and unchecking them. Make it so it looks like this, then select OK:
• Edit one of the cells in column H. You should see that the formula is gone, and there is only a number. Sometimes, after a Paste Special operation, when you edit a cell you may also see an apostrophe has been inserted into the number. This is an infamous bug. You can remove the apostrophes by selecting the column, going to Data → Text to Columns. Just select OK, and the problem will be fixed.
• Finally, format your column of numbers correctly. Select the column (or range H1:H417), right click on the selected area to bring up the secondary menu. Choose “Format Cells”. In Numbers, select the category ‘Numbers’, and choose -1234, and then change the value for Decimal places to read “1”. Then click OK.
• Now your numbers are ready to use.

Problem 5: Structural problems - data in inconvenient places

What’s the problem? Look closely at column F of your increasingly clean GRAIN dataset. This contains details about the intended use to which sold land will be put. Here are the first 10 entries (if your sheet is sorted alphabetically by ‘Landgrabbred’):

• Milk, olive oil, potatoes
• Rice
• Oil palm
• Oil palm
• Sugar cane
• Oilseed
• Rice
• Soybeans
• Maize, soybeans, wheat
• Barley, maize, soybeans, sunflower, wheat

In some cells there are single values, like just Oil palm. In others, the picture of how land is used is more complicated and there are more values. At a simple level, this data means we can do some basic analysis. Try this:

1. Let’s try and find all the land deals where Alfalfa was to be produced. Select the complete dataset. Go to Data → Filter → Autofilter. You’ll see the little triangles appear on the column headings, like so:

2. Select the little triangle, and a selection list will appear, which contains a list of all the values column F, listed alphabetically and without duplicates. Choosing from this list will change what data is shown in the spreadsheet:

3. In this list, untick Alfalfa. If you clicked “OK” now, the spreadsheet would show all rows of data that have everything but Alfalfa in column F. We can reverse this by unchecking Alfalfa and selecting the reverse button (at the bottom right of the window above you see two pictorial buttons, choose the left one). This reverses things, and shows only the unselected values. Click “OK”, and you will see only rows of data where the single word Alfalfa is present in column F. There are only two.
4. There are clearly other records where Alfalfa is recorded in Production. Repeat the above steps but include the 
items on the list that say “Alfalfa,crops,livestock” and “Alfalfa,maize,sunflower”. With this filter there are 2 
more rows of data, showing us 4 deals where Alfalfa was grown.

5. To remove the filter and again show all your data, go to Data → Filter → Remove Filter.

So we can do some useful basic stuff. But there are problems that will affect the analysis:

• We can’t see the complete range of options very easily.
• We have to rely on the people creating the data to have arranged things alphabetically too; what if someone had 
recorded Alfalfa at the end of the list? We couldn’t see it.
• Further, we assume they’ve spelled things the same each time, or used the same word for example: “Alfalfa 
crop” or perhaps another word for it.
• It’s difficult to get a full list of the land uses that you can look for.

What’s the solution? In a spreadsheet this can be partially overcome using the standard filter, which is more flexible 
than an autofilter.

Use standard filters for a more flexible search

1. Go to Data → Filter → Standard Filter. This opens up a window like the one below. It has a lot more options. 
   Let’s look for deals involving Alfalfa again. Make your standard filter look like this, and select OK:

2. So, this searches down Column F, where our data about production is stored, for cells where the word “alfalfa” 
is written anywhere. It doesn’t matter whether other words are there. The spreadsheet will again be filtered to 
show four rows.

3. We can build up more complicated queries. For example, try this one, which will filter the data for deals where 
the land use was thought to be for rice AND bananas, amongst others:

4. This returns only two results.

5. To remove the filter, go to Data → Filter → Remove.

Why this is only a part solution Again, it’s sort-of-useful but quite limited for the same reasons as autofilter: mis-
spellings, alternative wording, not having a complete list to choose from. At root this is a very difficult problem to 
solve with a spreadsheet: data on Production is what we call a repeatable field, in that buyers of land can have many 
equally important uses for the land. This is a different dimension of data: it’s called a one-to-many relationship. There 
is no easy way to structure data for a spreadsheet to make this data easy to use with any accuracy.

A common mis-step at this point is to start adding columns to allow multiple data to be recorded. This isn’t any better 
than recording it all in one cell, because of the way spreadsheet filters work. For example:

<table>
<thead>
<tr>
<th>Row</th>
<th>Production 1</th>
<th>Production 2</th>
<th>Production 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rice</td>
<td>Bananas</td>
<td>Grains</td>
</tr>
<tr>
<td>2</td>
<td>Grain</td>
<td>Rice</td>
<td>Bananas</td>
</tr>
<tr>
<td>3</td>
<td>Bananas</td>
<td>Grains</td>
<td>Rice</td>
</tr>
</tbody>
</table>

In the example above, all three rows have entries for “Rice”. A spreadsheet filter looks only up and down a column, not 
left and right to other columns. So to build a filter that accurately returned all three rows, you would need to search all 
three columns for the term “Rice”. This quickly becomes impractical as you begin wanting to find rows with different 
combinations of production types.

At this point, altering the structure of the GRAIN data for use in a spreadsheet probably isn’t worth it as there would 
be little gain. However, in this dataset, around 30% of entries in this column have more than a single entry, and there 
are nearly a 100 types of production. Having the data is important, as it may allow us to ask and answers questions
we wouldn’t otherwise be able to. For example, are there certain land uses that go together? Is there a relationship between the land use, the size of the land transacted, or the amount of investment. Exploiting the data effectively requires a database, not a spreadsheet.

However, there are things we can do to improve our ability to analyse the data, which we will go into in the next section.

Problem 6: from “banabas” to “bananas” - dealing with inconsistencies in data

What’s the problem? In the GRAIN dataset look again at column F, called ‘Production’. This column contains data about what buyers of land intend to grow on their new acquisitions, such as growing ‘cereal’, ‘soyabeans’ or ‘sugarcane’ and many other types of agricultural activity. As mentioned above, we can use this data to sort and filter the dataset, which helps us see the extent of different kinds of production. However, there are some uncertainties that make the data in the ‘Production’ column far less useful than it could be:

- We don’t know everything we could be looking for in the dataset. This is because there isn’t a single list of categories of production (called a ‘controlled vocabulary”). There are in fact well over 100 distinct categories. How do we get them into a single big list?
- We can’t be confident that the people creating the data have been consistent in how they entered the data. So if we look for “oil seed”, you’ll also find “oilsed” and “oil seeds”. This difference is important because a spreadsheet treats these these as completely different things.

We can address these obstacles to our understanding of the data. What we are going to do is create a new, more accurate and consistent controlled vocabulary for this dataset and apply it to the data. Here’s how.

What’s the solution? The process of solving this problem has four steps, which we’ll cover in detail:

1. Get all the categories that appear in the data
2. Make one big list from the almighty mess
3. Remove duplicates and correct categories that are nearly the same
4. Edit the data to fit the more accurate list of categories

Get all the categories that appear in the data As we discussed above in the section on structural problems, different pieces of data about production are held in a single cell. Each item is separated by commas. We can use a feature called “Text to Columns” to separate out each of the items into individual cells.

1. Create a new empty worksheet and name it “Text-to-columns” (or something memorable so you know what’s in it). We will use this worksheet as a ‘scratch pad’ to work on this particular problem.
2. In the GRAIN dataset that you have been working on, select Column F containing the data about production. Copy it (Shortcut: Ctrl-C) and then go to your new “Text-to-Columns” worksheet and paste it into column A. (You can use Paste Special if you like, to remove all the formatting). Rename the column as “Production (original)”
3. If you haven’t already, then you will now have to remove non-printing characters from this data. You can use the combination of CLEAN and TRIM and then Paste-Special that we described above. So, in cell B1 enter =CLEAN(TRIM(A1)). This will print out a version of the data in cell A1 that has no non-printing characters. Copy the formula down the column B to create a completely clean version of column A. Then select column B, copy it, and use Edit → Paste Special, unchecking “formulas”.
4. Now you have clean, usable data in column B. Rename it “Production (cleaned)”.

1.2. An introduction to the data pipeline 65
5. Now the cool bit: Select column B that has your cleaned data, and select **Data → Text to Columns**. A new window like this will appear:

   In “Separator options” area uncheck “tab” and check “comma”, as the items in our cells are separated by commas, and not tabs. In the “Fields” area you can see that for each comma the spreadsheet finds, it will move the data after that comma into a new column:

   Select **Okay**, and watch what happens. Here’s how your data now looks:

Do a few quick checks to see that nothing strange has happened. For example, check that the number of rows remains the same: there are 417 rows in the original data - this should not change. Also run your eye across a few of the rows to see things match.

**Make one big list from the almighty mess**  Why? This helps us find the unique items. There are over 700 individual items and many duplicates (to count them put =COUNTA(B2:P418) into a cell somewhere) and we can’t drill them down easily as they are currently structured. It’s easier if they are in a single column. To do this, we can use a little hack that occurs when you change file formats:

1. Save your spreadsheet. Now delete column A and save it again with a new filename. Now save it again (yawn!), this time as a ‘Text CSV’.

2. Start up your favourite text editor (like Notepad on Windows, Gedit on Linux, or TextEdit on Mac) and open the Text CSV you just created. It’ll look something like this when it’s opened:

3. Now copy and paste this text into a new document in **Libre Office Writer**. We will use Writer to process this data as text, before we return it to a spreadsheet for analysis. So, in Writer we will use the **Find and Replace** tool - it’s the same as in the spreadsheet - to create a single list:

4. Replace all the commas with new lines: open the Find and Replace feature. Make sure “Regular Expressions” are enabled (look in “More Options”). In Find enter a single comma. In Replace enter n. This will replace every comma with a new line. There will now be lots of blank lines - don’t fret, this doesn’t matter.

5. Select all the text (Ctrl-A) and copy it into the clipboard (Ctrl-C). Go back to your spreadsheet in **Libre Office Calc**. In a new worksheet select cell A1 and paste the text (Ctrl-V). The data will appear, but with lots of blank rows.

6. Select column A and go to **Data → Pivot Table**. Choose the options like this the below image.

7. In a new worksheet a list of unique terms used to describe Production, arranged alphabetically, along with a count of how many times they occur, will appear:

**Remove duplicates and correct categories that are nearly the same**  You can see immediately from the data above that there are many problems:

- “cofee” is mis-spelt, so is “forstry” (which should be ‘forestry’, which could be the same as ‘forest’)
- there are entries for “fruit” and “fruits”: which one is best?

This work is called ‘reconciliation’ and is a process designed to bring clarity to data. It involves looking through a list of terms and:

- identifying terms that mean the same thing and creating a new list
- applying the new list to the dataset.

We’ll go through these one by one.

**Bring the data into a form in which it’s easy to do this task**  

- Copy the list of items from the Pivot Table you made above and add it into a new worksheet. Use CLEAN and TRIM on it, and then sort it alphabetically in ascending order.
• Insert a row at the top. Label column A “Category”, Column B “Issue”, Column C “New Term”.

Identifying terms that mean the same thing

1. Go down the list and look for terms that mean the same thing. Here are some things that should alert your suspicion about a term:

• Spelling mistakes e.g. “bananas” vs “banabas”

• Differences in case: “fruit” vs “Fruit”. Choose your case and stick to it.

• Multiples: “fruits” vs “fruit”. Choose which one you want to use. Adjectives: “sweet sorghum”, “winter barley”. If there’s a similar category, like just “barley”, it may make sense to remove this more specific category.

• Additional terms: “and” in the text eg. “Dairy and Grain farms”; “Citrus and Olives”; “Crops (sorghum)”. The rule is to have only one category in each cell. So delete one of the terms and add it to the list on its own if it doesn’t exist.

• Qualifying terms eg “beef cattle” vs “beef”. “Crops” vs “food crops”. Choose which one.

1. In column B record what you think is the problem e.g. Near match, none, Spelling error. This will just help you keep track of the changes you make.

2. When you’ve gone down the list and identified the problems, then make the changes in column C. Here’s what we did (we also ringed the suspected duplicates as we went along):

1. Once you’ve done this, run Data → Pivot Table on your list of new terms (in Column C). You’ll see a huge difference.

2. By removing duplicates, spelling and grammar differences and so on we have cut down the categories from 149 to 88, which is still quite an extensive list! Anyhow, we have a more useful controlled vocabulary. The next step is to apply this to the data, so it can help us with our analysis.

Edit the data to fit the more accurate list of categories

So in order to create a more accurate analysis of the GRAIN dataset we have narrowed down the categories used to describe the sorts of land use to which transacted land is put. The last step is to bring the data into compliance with these new terms. We can do this using a combination of three useful features of the spreadsheet:

• Conditional formatting: we mentioned this ‘above’. It changes the formatting of a cell based on a rule that you give it eg. turn any cell in a given range red if it has the word “Sheep” in it. We can use this to highlight production categories that are not in our new improved list of categories.

• Data validation: this enables you to restrict what data is entered into a cell. So you specify a list of allowed values, and rather than type what you like into a cell you choose from a list. We can use this to make the changes to the data to bring it in line with the new categories, whilst reducing the risk of introducing more errors.

• Concatenation: this merges the contents of cells together. We will use it to put the improved data about production back together

The two things you will need are:

• the spreadsheet in which you used Text to Columns to separate out the terms used to describe land use

• your new, cleaned up list of terms taken from the previous pivot table.

Step 1: Find the items that need correcting We can use the spreadsheet to find the items that need correcting by comparing the data we have to the new list of categories. Here’s how:

1. In the worksheet where you have the data about production, make sure you have cleaned the data (using CLEAN and TRIM and paste special to remove the formulas).
2. Have the split items in a worksheet called “Split”. Place your new list of categories in a worksheet called “Categories”. Select your list of categories. Go to Data → Define Range. This window will appear, which will enable us to make the list of categories into a “range” against which we can make comparisons:

3. You’ve already selected the list of categories, which you can see displayed in the Range area. Type Production-Categories into the Name area and then select Add. We can now use this range.

4. Select all the data (Shortcut: Ctrl-A) in the worksheet called “Split”, where the data about production use is split across different columns. Go to Format → Conditional Formatting and make it look like the image below. The formula to use is COUNTIF(ProductionCategories,A1)=0. Also, select “New Style”. A new window called “Cell Style” will pop up. In the “Font Effects” tab choose a colour and then select okay. We chose red, which is shown below:

Select OK. This highlights in red the text in cells that are not found in the data range we have called “ProductionCategories”. The effect this has is to highlight the entries that we have to now correct. You spreadsheet will look like this:

Step 2: Correct these entries  Now we know where to look, we can make corrections to the data. The way to do this is to introduce data validation to the spreadsheet. This restricts the data that can be entered into a cell.

1. Select the complete dataset in the worksheet called ‘Split”, where you’ve just highlighted the values that don’t appear on the improved list of categories. Go to Data → Validity.

2. In the window that opens, make the fields look like those in the image below:

What this does is tell the spreadsheet that the only values that it should allow you to enter come from the list to be found in Cells A2:A88 of the worksheet called “Categories”. In others words: your list of categories. We also need to decide what to do when a value that isn’t on the list is entered. Select “Error Alert” and make it look like this to stop any non-list values being entered:

Click OK, and go to a cell with red text in it and click on it. You’ll see that a little drop down selector on the right hand side of the cell.

Click on it to display the list of ‘approved’ terms:

You can now go through the data, correcting it to remove the errors and make it more useful for analysis. There are a few things to watch out for:

- As you go through, increasingly you can use keyboard shortcuts and auto-complete and rely on the validation to tell you when you’ve typed a wrong entry.

- When you have changed a value, notice that the text changes to colour to show that it is now a recognised term. When there’s no more red, you’re done.

- With values like “Soyabean and other crops”, you should change it to “Soybean” and then add a new entry for “Crops”. Don’t forget!

Step 3: put it all back together again  We will take the improved data about land use and re-incorporate it into the full dataset.

- In you first worksheet, where you have been progressively cleaning the data, insert 15 columns to the right of Column F. Take the data from the worksheet that you validated the production category data in and copy it across into the new columns. If you didn’t put column heading, remember to paste starting in Row 2. You’ll be able to see the original data in Column F, and the separated about and cleaned data in Columns G to S.
• What we need to do now is put the clean data back into a single cell. We can do this with the CONCATENATE function. This allows you to take data from different cells and blend it into one. For example:

<table>
<thead>
<tr>
<th>row</th>
<th>F</th>
<th>G</th>
<th>Formula</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Cereal</td>
<td>Palm Oil</td>
<td>=CONCATENATE(F22,F22)</td>
<td>CerealPalm Oil</td>
</tr>
<tr>
<td>23</td>
<td>Cereal</td>
<td>Palm Oil</td>
<td>=CONCATENATE(F22,&quot; &quot;,G22)</td>
<td>Cereal Palm Oil</td>
</tr>
<tr>
<td>24</td>
<td>Cereal</td>
<td>Palm Oil</td>
<td>=CONCATENATE(F22,&quot; &quot;,G22)</td>
<td>Cereal Palm Oil</td>
</tr>
<tr>
<td>25</td>
<td>Cereal</td>
<td>Palm Oil</td>
<td>=CONCATENATE(F22,&quot; and &quot;,G22)</td>
<td>Cereal and Palm Oil</td>
</tr>
</tbody>
</table>

• We can blend data from other cells (which we called ‘referencing’), with other text to make it more readable. The examples above show how this works.

• Now let’s apply it to the data in our spreadsheet. In cell T2 insert the following formula:

<table>
<thead>
<tr>
<th>T2 (formula)</th>
<th>T2 (output)</th>
</tr>
</thead>
</table>

• It looks messy, but follow the logic. It looks across row 2 in columns G to S for data, and then prints it with a comma in between. We don’t know which cells have data in them precisely, so there is a list of trailing commas which print where a cell is empty. We can get rid of these easily with the LEFT and SEARCH formulas we used ‘above’. In cell U2:

<table>
<thead>
<tr>
<th>T2 (output)</th>
<th>U2 (formula)</th>
<th>U2 (Output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milk, Olives, potatoes, , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,</td>
<td>=LEFT(T2,SEARCH(“,&quot;,T2)-1)</td>
<td>Milk, Olives, potatoes</td>
</tr>
</tbody>
</table>

• So, with these two formulas in place, they can be copied down the spreadsheet to complete the operation. The only row that this doesn’t work for is row 94, Temasek’s purchase of land in China. You can correct it by hand :)

• The final step is to make the new data usable. If you’re confident the work is done, insert column to the left of the original column F called Production. Call this new Column G “Production (cleaned)”. Select all the data in Column T (rows 2 to 417), move the cursor and use Paste Special to insert it into Column G.

• You can now either delete or hide columns H to W in which you’ve been working.

**Step 4: using the cleaned data about types of production land use** You can repeat the steps that we outlined in Problem 4 above, using the standard filters to build queries on the cleaned data in column G. This time you will not have problems with inconsistencies, mis-spellings and so on.

**Finishing touches**

• Columns A and C of the GRAIN data both record names of states. To check for errors here we copied the data from both columns into a single column, and created a pivot table that showed the list of unique values recorded in both columns. The only errors were a mis-spelling of Germany (“Gemany”) and the use of “West Africa”, which is a region not a state. We corrected both, including a note in the “Summary” cell if further clarity was needed.

• To complete this, apply the cleaning process outlined in Problem 5 above to Column D of the original GRAIN dataset, which records data about the sector. But for some issues with lower and upper case it does not share the same problem as the data about Production in column F. There are not obviously overlapping categories. To be certain, we ran CLEAN and TRIM again, and converted everything to sentence case (the formula is =PROPER(CLEAN(TRIM(cell reference))).

• In Column H (Status of Deal), we altered three entries that had additional text not in the categories Done, In Process, Proposed, Suspended, MOU signed. The information carried by this text was already continued in the Summary (column I).
Cleaning Data with Refine

What you’ll need:

1. Refine - Download it from openrefine.org
2. The sample Dataset - Download it from Africa Open Data

Step 1: Creating a new Project

Open Refine (previously Google Refine) is a data cleaning software that uses your web browser as an interface. This means it will look like it runs on the internet but all your data remains on your machine and you do not need internet connection to work with it.

The main aim of Refine is to help you exploring and cleaning your data before you use it further. It is built for large datasets - so don’t worry as long as your spreadsheets can keep the information: Refine can as well.

To work with your data in Refine you need to start a new project:

Walkthrough: Creating a Refine project

1. Start Refine - this will open a browser window pointing to http://127.0.0.1:3333 if this doesn’t happen open the link with your browser directly
2. Create a new project: On the left tab select the “Create Project” tab:
3. Click on “Choose Files” to choose your downloaded file and click on “next” - you can also use the URL to the CSV directly if your data is hosted on the web.
4. You will get a preview on how refine will interpret your data - if you have selected a well formatted CSV or other file: this should be pretty automatic.
5. Review the preview carefully to make sure the data looks right. Double check character encoding. Much, but not all data uses UTF-8 these days, but make sure you don’t see any funny characters in preview.
6. You may want to turn of “guess data types”, particularly if you have data that contains leading zeros in numbers or identifiers which are significant.
7. Name your project in the box on the top right side and click on “Create Project”
8. The project will open in the project view, this is the basic interface you are going to work with: by default refine shows only 10 rows of data, you can change this on the bar above the data rows. Also you can use the navigation on the right to see the next or previous rows.

You now have successfully created your first Refine project. Remember: although it runs in a web-browser, the Refine server is still on your machine - all the data is there (so no worries if you handle sensitive information)

Step 2: Sorting and Facetting

Once we created our project, let’s go and explore the data and the Refine interface a bit. Using Refine might be intimidating at first, since it seems so different from spreadsheets, once you get used to it you will notice how easily you can do things with it.

One of the commonly used functions in spreadsheets is sorting and filtering data - to figure out minima, maxima or things about certain categories. Refine can do the same thing.

Walkthrough: Sorting rows

1. Refines handles data similar to a spreadsheet: you have rows, columns and cells - a cell is a field defined by a row and a column.
2. To sort your rows based on a specific column click on the small downward triangle next to the column
3. Select “Sort...” to open the sorting dialog
4. You can select what to sort the values as and then what order to sort in. (We’ll sort in text, since for now we only have text columns)

5. Click “OK” and your rows will be sorted based on the column

6. To undo the sort, click on the column options again, select “sort” then “remove sort”

The other frequently used function in Spreadsheets is filtering - in Refine this is called facetting. Facetting in Refine is really powerful - you will see in most of the rest of the Recipe we’ll use facets.

**Walkthrough: Facetting rows based on a column**

1. Select the column options for the column you want to facet with

2. Select “Facet”

3. You can facet differently for text, numbers or dates - let’s facet as text - click on “Text facet”

4. This will open a facet in the left bar

5. Now select one or more of the choices and you’ll see how your data rows are limited to just those selected.

6. Of course you can add more than one facet and thus filter more than once.

**Step 3 Dealing with Blank Cells** If you look closely at your facets, you'll notice that on the bottom you have a selector saying “(blank)” for this - we’ll need to deal with it.

**Walkthrough: Filling in the (blank)s**

1. Choose the “(blank)” facet in your “Owner” column

2. If you look at some of the rows, you’ll see that there was a mis-split of the columns and the owner actually ended up in the “Category” Column

3. To fill this into the “Owner” Field hover over the cell you want to fill in and click the “edit” button.

4. If you click the “Edit” button you can add the Owner there - don’t forget to also correct the “Category” cell.

5. You’ll notice some rows seem to be erratic - they don’t have a name that makes sense and no further information - you can flag these for deletion by clicking on the little flag.

6. Do the same with the “Category” Column - the Category is sometimes joined with the “Name” column

7. Now let’s delete the flagged rows - make sure you are in row mode for this: for this click on “row” in the top left corner above the data.

8. Open the column options for “All” and select “Facet” - “Facet by Flag”

9. Now you can select “true” in your flag facet on the left.

10. Now let’s delete the flagged rows: in the Column options for “All” select “Edit rows” - “Remove all matching rows”

**Step 4: Fighting the Invisible Man** As illustrated in The Invisible Man is in your Spreadsheets having spaces or newlines in your datafields is a problem. Since this is a very common problem, Refine has specific functions to remove whitespaces that shouldn’t be there.

**Walkthrough: Removing hidden whitespaces**

1. Let’s start cleaning our Dataset with the Owner Column

2. Create a Text Facet for the Owner Column as described above
3. You will notice that there are several things odd in the column: It starts with a long list of similar looking entries - we’ll deal with it later.
   Although they look similar to you, they are different for the computer - there is a different number of spaces between the quotes.
4. Scroll down and you’ll notice that some entries will be there twice - although they look similar. There are two entries for Municipality that look exactly the same. This is because they have whitespaces at the end.
5. Refine can help you clean this up in an instant - open the column options for the “Owner” column
7. This will remove whitespaces in the beginning and at the end of your column
8. Check Municipality and you’ll note that there’s only one choice now - perfect. Now let’s deal with the list at the beginning.
10. You’ll see the multiple choices have been reduced to two choices in an instant
11. Now our list already looks a lot cleaner!
12. Go ahead and apply the two transforms to all your columns.

Once you made your transforms you might wonder: What if I made a mistake? Also if you work with data you generally want to keep track of what you did to the data. Since Refine was build with data processing in mind it keeps track of what you’re doing with your data and allows you to go back and forth in time. To see your history of changes click on the “Undo/Redo” tab on the left.

You see all the changes you made - by simply clicking on one of the steps you’ll be undoing all the changes after the step (don’t worry you can redo pretty much the same way). Play with this system until you are comfortable.

**Step 5. Reconciling categories**  A quick look at our categories and you’ll notice that not everything is well in Owner land - still some categories that should be the same are not. The same for the “Category” column - let’s reconcile them.

**Walkthrough: Reconciling Categories**

1. Create a Facet for the column you want to reconcile (in our case this is “Owner”)
2. The first step is to bring the categories to the same case - see for example “Town Council” and “Town council” - the difference is just one letter.
3. Refine can help you to automatically find the categories that belong together - a feature it calls “Clustering”. To activate clustering click on the “Cluster” button in your facet.
4. You will end up in the clustering menu - as you can see Refine is pretty smart about which things should belong together
5. Check the “merge” checkbox if you want the two categories to be the same. Once you marked all the categories you want to merge click on “Merge selected & Re-Cluster”
6. If Refine doesn’t find more values to be similar change the “Keying Function” and see whether you can find more similar categories - if not: simply click close to continue.
7. This reconciled some of your values - let’s go on.
8. Look at “Mission” for example we have three different categories for what should be one - Refine did not automatically find them.
9. Let’s change them all to mission
10. Hover over “Mission Hosp.” notice the “edit” button at the end?
11. Click on Edit - this will open the field for editing. Change the name to “Mission” - this will change “Mission Hosp.” to “Mission” in all cells where it appears - continue on to change all the fields you can find.

12. Repeat reconciling for “Category”

Step 6: Splitting Columns If you look at the “Name” column in our dataset you’ll notice that the names commonly start with a number (this is an enumeration of hospitals in a district - and is an artifact from extracting the data). Let’s clean this up and split the number and the name.

Walkthrough: Splitting Columns

1. To Split a column select “Edit Column” “Split into several columns”
2. We want to split at a “.” since the number generally ends with a “.”
3. Enter “.” into the Seperator for in the split menu - since we only want to have two new columns enter 2 into the field below so the sentence reads “Split into 2 columns at most”
4. Click on “OK” and you’ll end up with two columns.
5. On some of the rows the split will fail - to fix those, create a facet on the second column and select “(blank)”
6. You can now manually fix the cells.

Congratulations - You successfully cleaned up a dataset using Refine!

However there is even more you can do with Refine: For example did you notice how there is always a number next to the categories in the facet - telling you how many rows are in that category? By combining two facets, can you find out how many clinics the government owns? And who owns the Provincial Hospitals?

Creating Line Charts

This tutorial uses Google spreadsheets to create a line chart.

There is sample data for this tutorial here .

So let’s create a line chart. Let’s say we want to see how healthcare expenditure evolved in Luxembourg, our top spending country.

1. Go back to your World Bank data sheet.
2. Remove the filter for years and filter for a single country: Luxembourg (you can do so by clicking the “clear” label in the filtering menu - then type Luxem and you’ll see Luxembourg appear - select it).
3. Now the only thing that’s left is sorting by years - so you have them in right order.
4. Select all of it and copy it to a new sheet.
5. Now move the columns year and “healthcare expenditure total per person” next to each other.
6. Select both columns and select “Chart...” from the “insert” menu.
7. Click on the “charts” tab and select the line chart you want to plot.
8. You already know how to manipulate the look of a chart, so go and play around until it looks similar to the chart above!

Walkthrough: Scatterplot

This tutorial uses Google spreadsheets to create a scatterplot. There is sample data for this tutorial ‘here‘ .

So let’s create a scatterplot.
1. Start with World Bank Data.

2. Copy it to a new sheet and put the columns “healthcare expenditure total per person” and “life expectancy” next to each other.

3. Click insert charts... and select “scatter plot” from charts.

4. Select the first one, since this is what we want to do.

5. And there you go: simply adapt the scatterplot so it looks nice. Don’t forget to label axes. Try to make the dots smaller if there is significant overlap.

Creating a Choropleth map

This tutorial uses Google spreadsheets to create a choropleth map. There is sample data for this tutorial ‘here’.

1. Filter for a single year (e.g. 2009) insert a new sheet and copy the filtered data into it.

2. As with all previous charts also here the columns need to be in a special position.

3. Move your data column (the one you want to use to display) right next to the country names.

4. Now mark the two columns and select “Chart...” from “insert”.

5. Under “Charts” select “Map” and then “geo chart - regions”.

6. You’ll see a preview. Play with the settings in customize to change the map, the colour-scale etc.

7. A note on colours: the red-green scale that is selected by default is not the best scale. So select a different one showing contrasts nicely.

Now let’s go ahead and publish our results on a webpage. This will get a bit techy but don’t worry we’ll guide you through. We will create a small web page containing some of the visualizations we created using a simple online tool called pastehtml. Pastehtml allows you to create html websites easily by simply editing the text online and then saving it for sharing.

Walkthrough: Presenting our information as a webpage

1. To start a webpage simply visit pastehtml.com

2. See the input box with all the brackets? This is html code and we’ll be editing it to present your results. (If you want to learn more about html code head to the school of webcraft

3. First let’s change the title of the webpage: This is the bit between `<title>” and “</title>”. Edit it so it is appropriate.

4. Then let’s go and edit the content for a webpage (this is the part between `<body>” and “</body>”). See the text between `<p>` this defines a paragraph. Go ahead and edit it!

5. You can click on “Publish page” to see how your page will look like (approximately).

6. On the top you’ll always have the possibility to go back and edit.

7. Now let’s add some charts we made.

8. Go back to one of the charts in the spreadsheet.

9. Click on the chart. See the small triangle top right of the chart: this is the options menu.

10. Go and select “Publish chart’85”. There will be a popup with a lot of code in a grey box:
11. Copy this code and paste it into the pastehtml (somewhere between <body> and </body>). Now if you go and look at your page, the chart should be there.

12. Once you are finished, click on publish and you’ll get a url to your webpage. use this to share your results with your friends.

Of course if you already have a blog or something similar you can share the results there

... raw:: html

    <div class="alert alert-info">Any questions? Got stuck? <a class="btn btn-large btn-info" href="http://ask.schoolofdata.org">Ask School of Data! </a></div>

Appendix

Glossary

**Anonymisation**  The process of treating data such that it cannot be used for the identification of individuals.

**API**  See *Application Programming Interface*.

**Application Programming Interface**  A way computer programmes talk to one another. Can be understood in terms of how a programmer sends instructions between programmes.

**Attribution Licence**  A licence that requires attributing the original source of the licensed material.

**BitTorrent**  BitTorrent is a protocol for distributing the bandwidth for transferring very large files between the computers which are participating in the transfer. Rather than downloading a file from a specific source, BitTorrent allows peers to download from each other.

**Boolean logic**  A form of algebra in which all values are reduced to either TRUE or FALSE.

**Categorical Data**  Data that helps put things into categories. E.g.: Country names, Groups, Conditions, Tags

**Choropleth Map**  A choropleth map is a map where value are encoded onto regions using colormapping. The whole region is colored using the underlying value.

**Comma-separated Values**  See *CSV*.

**Continuous Data**  Numerical data that, if you plot all possible values, has no gaps. E.g. Sizes (you can be 155.55 or 155.56cm tall etc.) Compare to *Discrete Data*

**Crowdsourcing**  Mashup of crowd and outsourcing: Having a lot of people do simple tasks to complete the whole work.

**CSV**  Comma Separated Values. A very simple, open format for tabular data which can be exported and imported by all spreadsheet applications and is easily manipulable with command line tools.

**curl**  [http://curl.haxx.se/](http://curl.haxx.se/) - a command line tool for transferring data to and from online systems over standard internet protocols including FTP and HTTP. Very powerful and great for working with *Web API*s from the command line.

**DAP**  See *Data Access Protocol*.

**Data Access Protocol**  A system that allows outsiders to be granted access to databases without overloading either system.

**Discrete Data**  Numerical Data that, if you plot all possible values, has gaps in it. E.g. the count of things (there are no 1.5 children). Compare to *Continuous Data*

**etherpad**  A piece of software for collaborative real-time editing of text. See [http://etherpad.org/](http://etherpad.org/).
GDP  Gross domestic product (GDP) is the market value of all officially recognized goods and services produced within a country in a given period of time. GDP per capita is often considered an indicator of a country’s standard of living. (Source: Wikipedia.)

Geocode  see Geocoding

Geocoding  From Geographical Coding. Describes the practice of attaching geographical coordinates to items.

GeoJSON  GeoJSON is a format for encoding a variety of geographic data structures. It is based on the JSON specification. More documentation can be found on http://www.geojson.org

Intellectual property rights  Monopolies granted to individuals for intellectual creations.

IP rights  See Intellectual property rights.

JSON  JavaScript Object Notation. A common format to exchange data. Although it is derived from Javascript, libraries to parse JSON data exist for many programming languages. Its compact style and ease of use has made it widespread. To make viewing JSON in a browser easier you can install a plugin such as JSONView in Chrome and JSONView in Firefox.

Machine-readable  Formats that are machine readable are ones which are able to have their data extracted by computer programs easily. PDF documents are not machine readable. Computers can display the text nicely, but have great difficulty understanding the context that surrounds the text. Common machine-readable file formats are CSV and Excel Files.

Mean  The arithmetic mean of a set of values. Calculated by summing up all values and then dividing by the number of values.

Median  The median is defined as the value where 50% of values in a range will be below, 50% of values above the value.

Normal Distribution  The normal (or Gaussian) distribution is a continuous probability distribution with a bell shaped curve.

Open Data  Open data is data that can be used, reused and redistributed freely by anyone for any purpose. More details can be found at at opendefinition.org.

Open standards  Generally understood as technical standards which are free from licencing restrictions. Can also be interpreted to mean standards which are developed in a vendor-neutral manner.

Percentiles  Percentiles are a value where n% of values are below in a given range. e.g. the 5th percentile: 5 percent of values are lower than this value.

Public domain  No copyright exists over the work. Does not exist in all jurisdictions.

Qualitative Data  Qualitative data is data telling you something about qualities: e.g. description, colors etc. Interviews count as qualitative data

Quantitative Data  Quantitative data tells you something about a measure or quantification. Such as the quantity of things you have, the size (if measured) etc.

Quartiles  Quartiles are the values where 25, 50 and 75% of values in a range are below the given value.

Readme  A file (usually named README or README.txt) that explains new users what the current directory or set of files is about. This is very commonly found in open source software projects and is considered good practice to be included with various publications (including datasets). The file usually contains a short description of what to expect.

Scraping  The process of extracting data in machine-readable formats of non-pure data sources e.g.: webpages or PDF documents. Often prefixed with the source (web-scraping PDF-scraping).

Share-alike Licence  A licence that requires users of a work to provide the content under the same or similar conditions as the original.
Tab-separated values  Tab-separated values (TSV) are a very common form of text file format for sharing tabular data. The format is extremely simple and highly *machine-readable*.

Taxonomy  Classification. Taxonomy refers to hierarchical classification of things. One of the best known is the Linnean classification of species - still used today to classify all living beings.

Web API  An *API* that is designed to work over the Internet.