Cleaning data (part 2)

Advanced data cleaning with OpenRefine
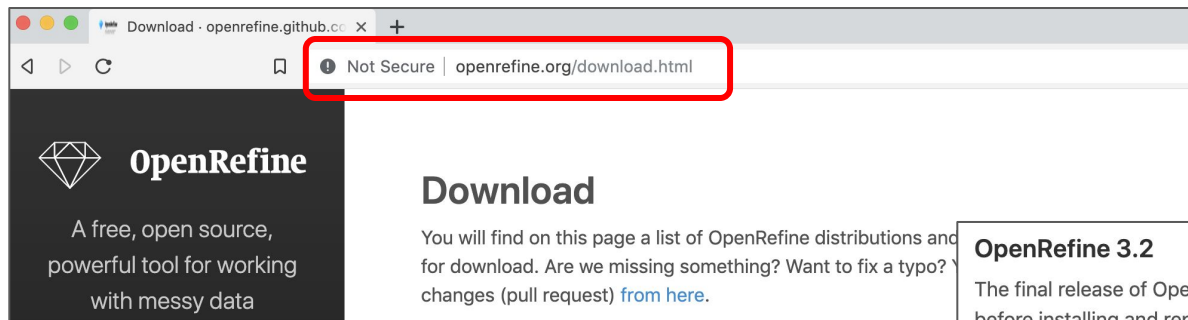
Martin Frigaard, MA, MAS

link for slides: http://bit.ly/bu-cleaning-data-p2

First steps:

Make sure you've downloaded and installed OpenRefine:

http://openrefine.org/download.html

1) Go to the Downloads page

2) Select the appropriate application for your computer

What we'll cover

- First things first (software installation check)

- Web technologies and non-rectangular data types

- Wrectangling your data

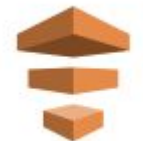- Make it reproducible

- Share your work!
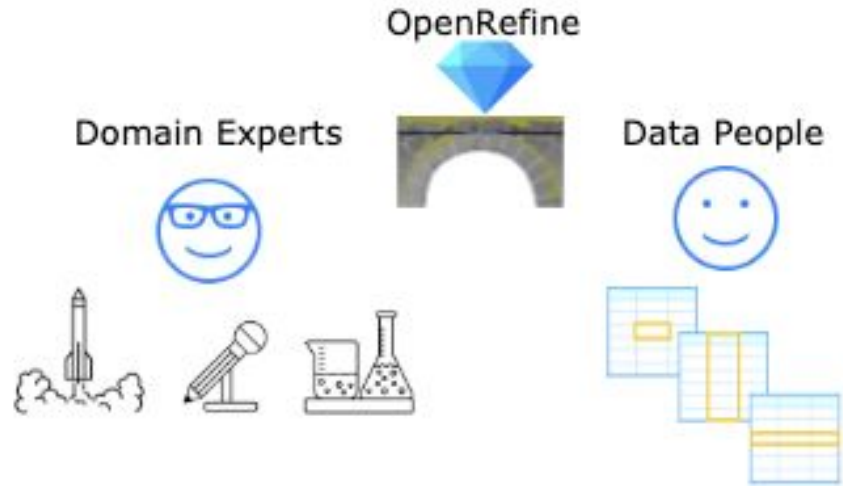
So much data work to do...

What is/where are the data?

Data are everywhere we look, and they're being used to measure nearly anything we can imagine
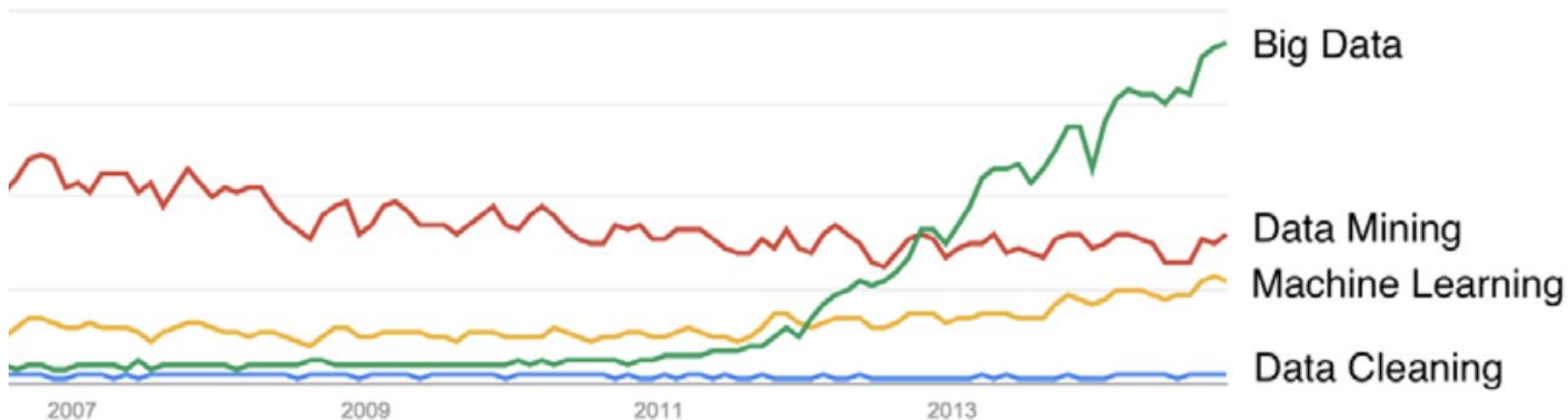
Numbers, text, pictures, audio, videos--there's no limit to what we can capture

But 'data cleaner' and 'data janitor' aren't very sexy...

Popularity of "Data" Terms on Google Trends, 2007-2014

Big Data

Data Mining

Machine Learning

Data Cleaning

2007   2009   2011   2013

Fundamentals – Data File Types & Structures

Data come in a variety of file formats

1. Someone sends you data in a downloadable file

2. You log into an interactive front-end and retrieve the data from a storage system (i.e. a MS SQL server database system)

3. The data are available via a continuous stream that's capturing web traffic (i.e. social media)

4. You access the data through an **Application Programming Interface (API)**
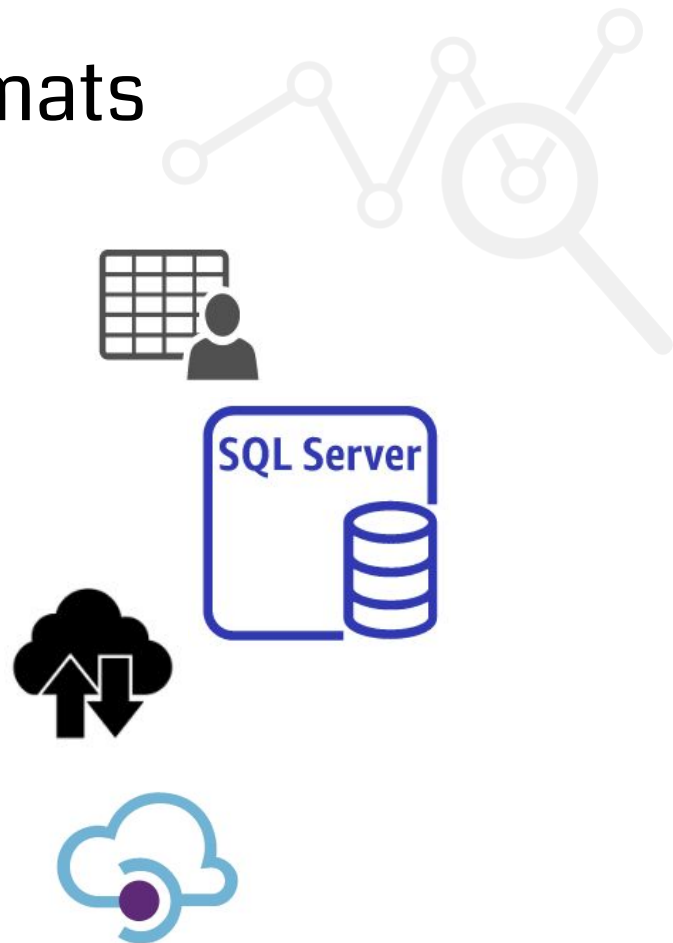
Types of files: text files

Computer files generally belong to two broad groups, commonly called **text files** and **binary files**

- Files like web pages, computer source code, and open-source programming languages are all text files
- These files can opened in a text editor (Notepad, Text Edit, etc.) or via the command line
- Humans and computers can read these files

Types of files: binary files

Binary files require software to open them, and are typically not human readable

- MS Word, Excel, and proprietary software files

- Executables and application installation files (**.dmg** or **.exe**)

- Media files (**.png**, **.jpg** **.mp4** or **.mov** files)

- Encryption or compression files (**.zip** or **.rar**)

- _Humans can't read these files!!_

Rectangular files (spreadsheets)

- These are files stored in columns and rows (or variables and observations)

- Rectangular files usually require spreadsheet software (Excel or OpenOffice) or relational database software
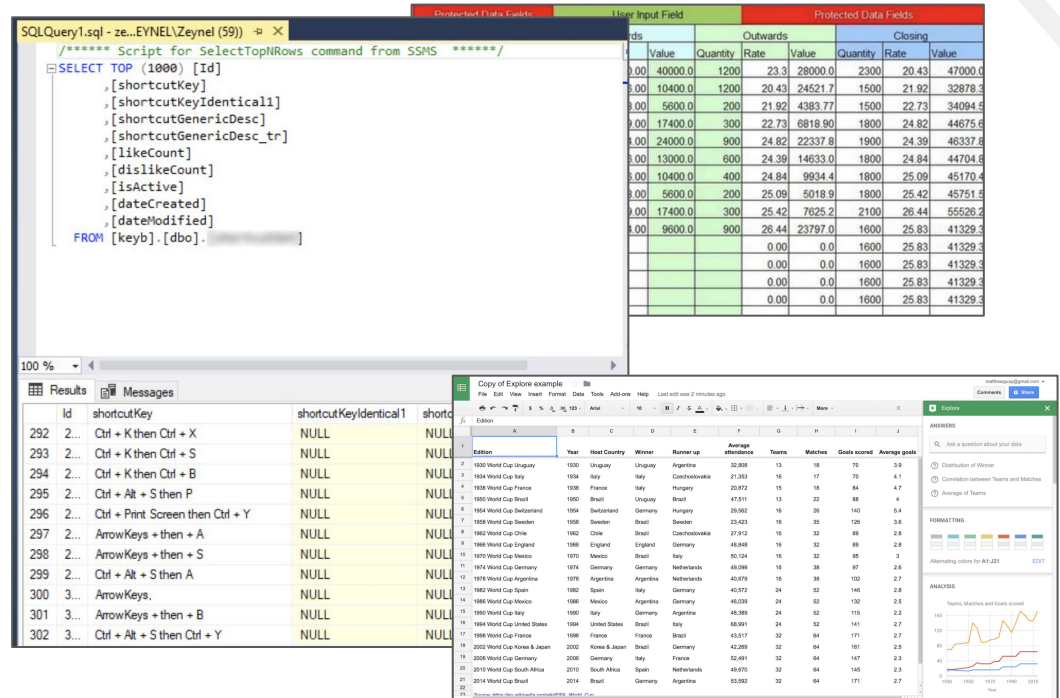
Non-rectangular files (JSON & XML)

- Non-rectangular data files are commonly used on the web for storing and transferring data

- **JSON** (JavaScript Object Notation) was created in 2002 and used for data storage and transfer

- **XML** (extensible markup language) is slightly older technology (created in 1996), but still widely used for the same purpose

JSON data objects

- JSON is an **object notation language** and stores data in **objects and arrays**

- *Why would anyone store data this way?* JSON can store data and the attributes about the data within the same object.
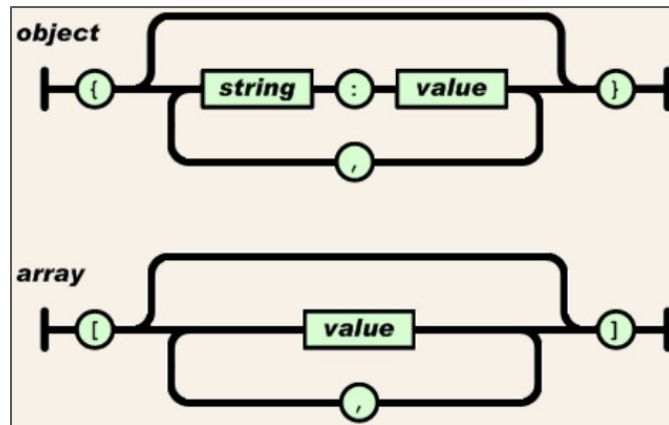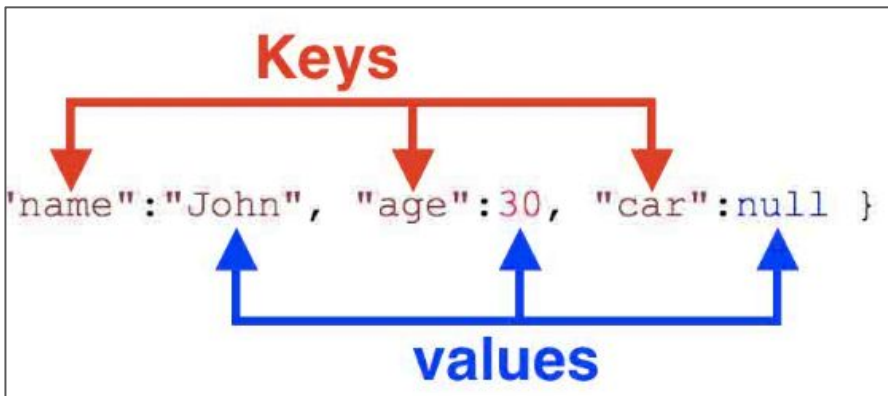
XML data objects

- **XML** (eXtensible Markup Language) is a language used to encode web documents and data structures

- XML is also useful for transmitting information between different software systems and through APIs

- XML was designed to be "self-descriptive" and carry data in a readable format

```xml
<?xml version="1.0"?>
<quiz>
 <qanda seq="1">
  <question>
   Who was the forty-second
   president of the U.S.A.?
  </question>
  <answer>
   William Jefferson Clinton
  </answer>
 </qanda>
 <!-- Note: We need to add
  more questions later.-->
</quiz>
```

XML

Why bother with these technologies?

API: Application Programming Interface

An [API](#) is the set of instructions for accessing or transmitting information between software applications

JSON and **XML** are usually how data are transferred in/out of APIs

API: Application Programming Interface

Applications capture images with meta data

These pictures are stored in an **image API** that allows search by photo contents.

api

MP4
JPG
PNG

Someone writes a query to the **image API** for pictures with cats and meta data (location)
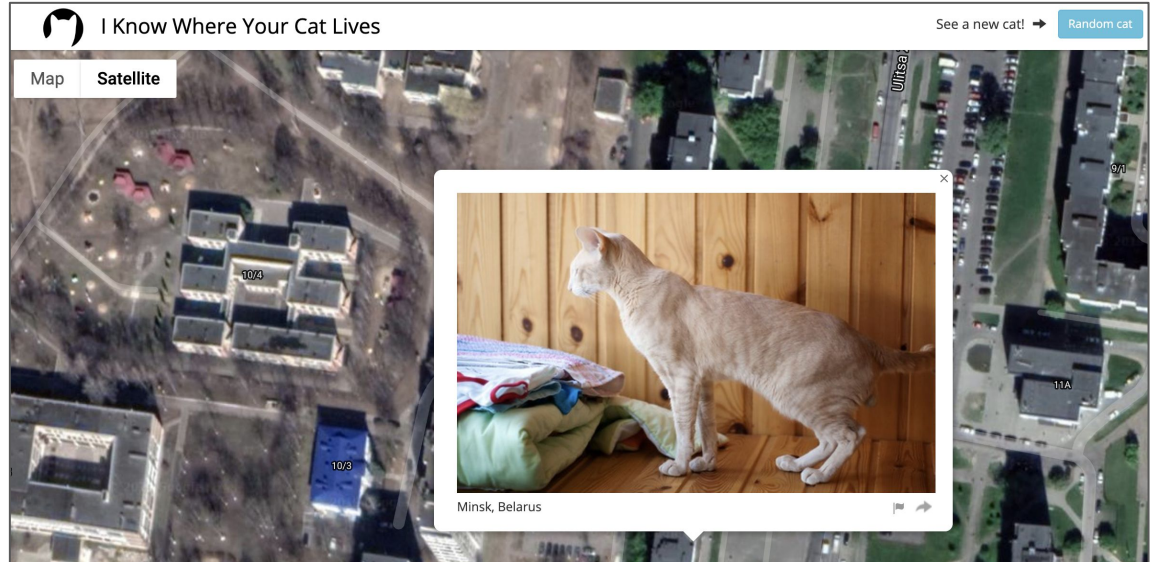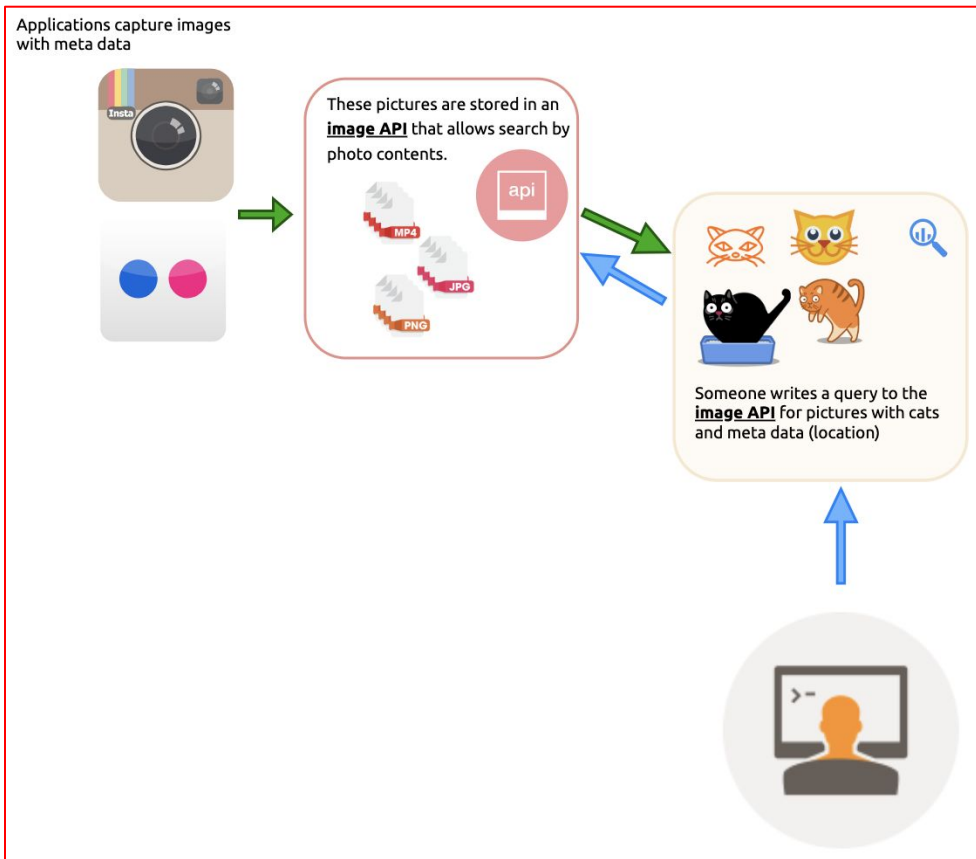
1) Photos from Flickr and Instagram get stored in an API

2) Requests are made for certain photos with GPS data

API: Application Programming Interface

Applications capture images with meta data

These pictures are stored in an **image API** that allows search by photo contents.

Someone writes a query to the **image API** for pictures with cats and meta data (location)

Open Street Map

api

Another API has GPS data (latitude and longitude) that link to the meta data in the photos

3) Combine image meta data with GPS data from OpenStreetMap API
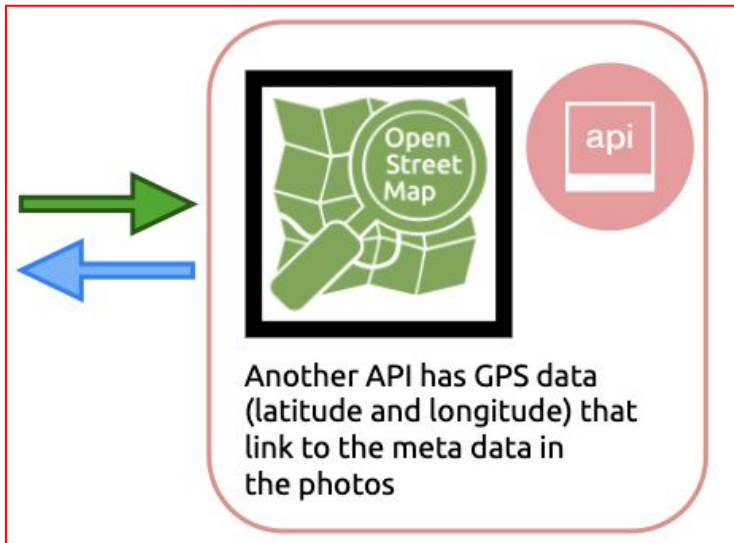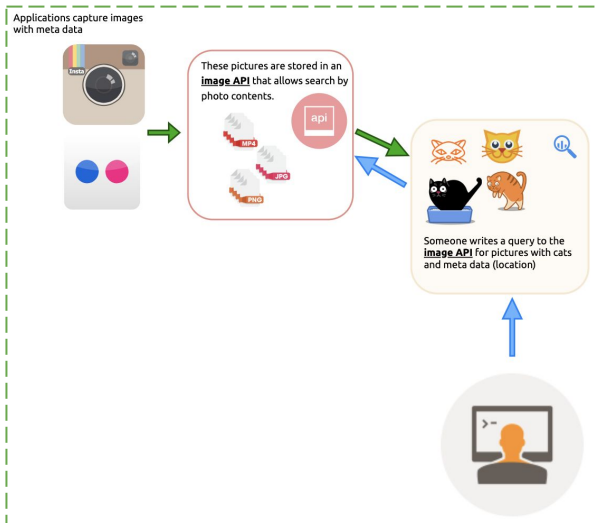
API: Application Programming Interface
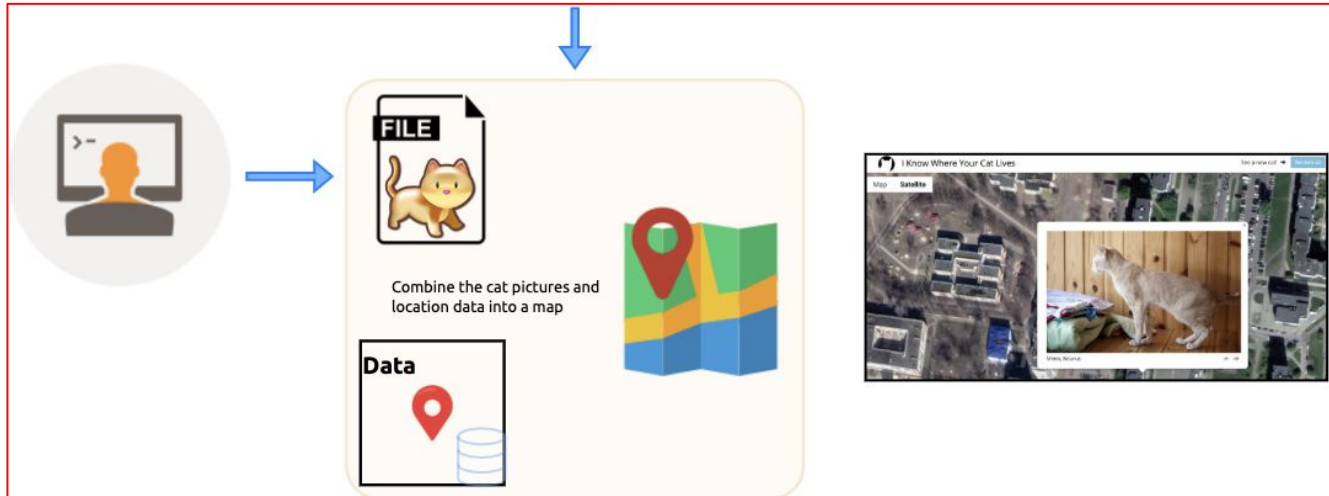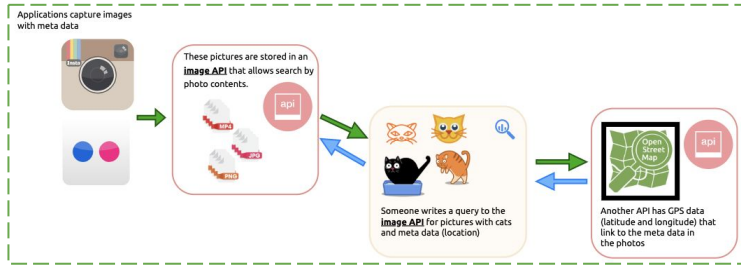
4) Use Google Maps to display the image on the satellite image

A Quick Example

Load Address Data into OpenRefine

1. Navigate to this link and load the addresses into OpenRefine:

Raw CSV address data

2. Create a new project called **us-addresses**

Create `full_address` column

Select the little arrow on the **street** column and click on,

Edit column >
  Add column based on this column

Use the following GREL code to create a new **full_address** column:

`value + ", " + cells["city_state_zip"].value`

Click **OK.**

Arrange columns

Select the little arrow on the new **full_address** column and click on,

Edit column > Move column to the end

MapQuest API

Head over to:

https://developer.mapquest.com/

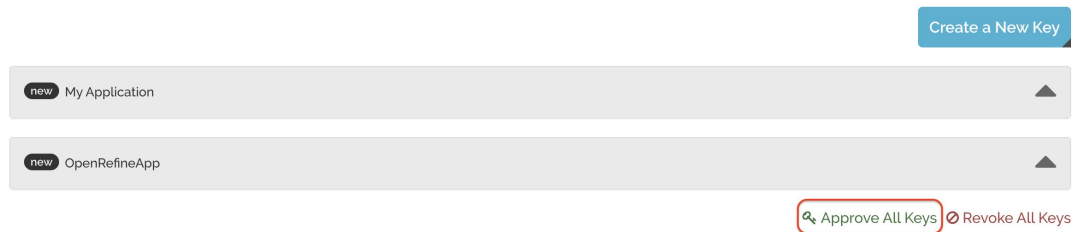Click on the button to get your **Free API Key**
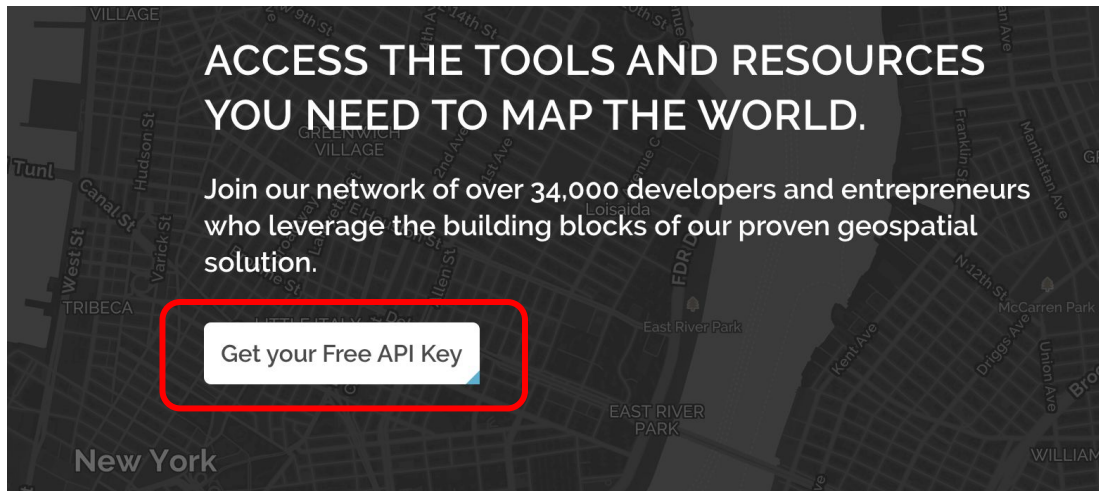
Create and Approve MapQuest API Key

1. You can create a new key or use the one listed under **My Application**
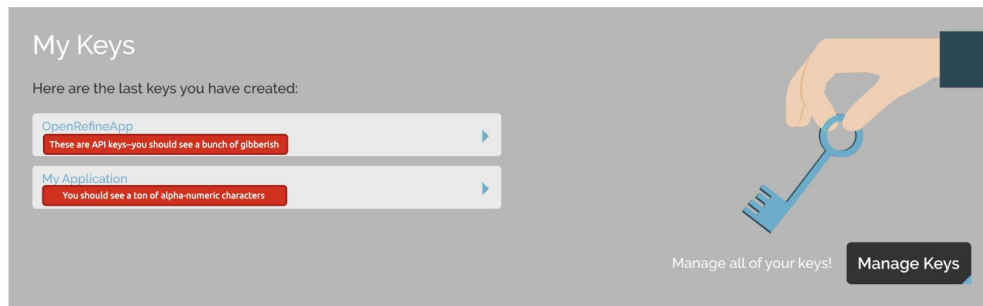
2. Click **Manage Keys** then **Approve All Keys**

APIs with OpenRefine

Example MapQuest API query

The domain address, proceeded by some additional information to access the server:

open.mapquestapi.com/nominatim/v1/

The request, written in **http** or hypertext transfer protocol:

http://

The specific resources on the server:
1) the (**search.php**) API, followed by '**?**'
2) data requests (**format=json**), followed by "**&**".
3) Our API key (**key=yourkey**), followed by "**&**"
4) The query (**q=[Addresses]**)

http:// open.mapquestapi.com/nominatim/v1/ search.php?format=json&key=JWbcrLmu5UfJ9n1krjAMdr0jz3QIG0ha&q=[Addresses]

Domain **&** directory

Example API query parameters with GREL

Click on the new full_address column and select, **Add column by fetching URLs**

Follow the steps in the diagram to the right, but refer to the notes below for the GREL expression:

After you've entered the **Expression**, check the 1st row in the **Preview** section at the bottom of the dialogue box.

Add column by fetching URLs based on column full_address

| New column name | mapquest_locations | Throttle delay | 1000 milliseconds |
| On error | ○ set to blank ● store error | ☑ Cache responses |

HTTP headers to be used when fetching URLs: Show

Formulate the URLs to fetch:

Expression   Language General Refine Expression Language (GREL)

```
'http://open.mapquestapi.com/nominatim/v1/search.php?'+ 'format=json&' +
'key=JWbcrLmu5UfJ9n1krjAMdr0jz3QIG0ha&'+ 'q='+ escape(value, "url")
```
No syntax error.

Preview   History   Starred   Help

row	value	'http://open.mapquestapi.com/n ...
1.	777 Brockton Avenue, Abington MA 2351	http://open.mapquestapi.com/nominatim/v1/search. format=json&key=JWbcrLmu5UfJ9n1krjAMdr0jz3Ql
2.	30 Memorial Drive, Avon MA 2322	http://open.mapquestapi.com/nominatim/v1/search. format=json&key=JWbcrLmu5UfJ9n1krjAMdr0jz3Ql
3.	250 Hartford Avenue, Bellingham MA 2019	http://open.mapquestapi.com/nominatim/v1/search. format=json&key=JWbcrLmu5UfJ9n1krjAMdr0jz3Ql
4.	700 Oak Street, Brockton MA 2301	http://open.mapquestapi.com/nominatim/v1/search. format=json&key=JWbcrLmu5UfJ9n1krjAMdr0jz3Ql

OK   Cancel

Example API query parameters with GREL

This takes time!

Create column mapquest_locations at index 3 by fetching
URLs based on column full_address using expression
grel:'http://open.mapquestapi.com/nominatim/v1/search.php?'+
'format=json&' +
'key=JWbcrLmu5UfJ9n1krjAMdr0jz3QlG0ha&'+ 'q='+
escape(value, "url")
78% complete   Cancel

JSON data

When you're done, you should see the following JSON data in the new **mapquest_locations** column

Parsing JSON data in OpenRefine

Click on the new `mapquest_locations` column and select **_Edit column > Add column based on this column..._**

In the dialogue box, enter the following options:

1. **Name new column** = `latitude`

2. **On error** = `store error`

3. **Expression** = `value.parseJson()[0].lat`

3. Verify this is correct in the **Preview** pane

Parsing JSON data in OpenRefine

Add column based on column mapquest_locations

New column name: longitude

On error: ○ set to blank ● store error ○ copy value from original column

Expression:
```
value.parseJson()[0].lon
```

No syntax error.

Language: General Refine Expression Language (GREL)

Preview | History | Starred | Help

row	value	value.parseJson()[0].lon
1.	[{"place_id":"117715864","licence":"Data © OpenStreetMap contributors, ODbL 1.0. https:\/\/www.openstreetmap.org\/copyright","osm_type":"way","c ["42.095747","42.0968228","-70.9694556","-70.967681"],"lat":"4: 777, Brockton Avenue, Abington, Plymouth County, Massachusetts, 02351, United States of America","class":"shop","type":"supermarket","importance":0.62` 10-116-146-77.mq-us-west-2.ec2.aolcloud.net\/nominatim\/images\/mapicons\/shopping_su	-70.9685309348312
2.	[{"place_id":"117673610","licence":"Data © OpenStreetMap	-71.0300878730199

OK   Cancel

Repeat the same steps on the `mapquest_locations` column, but this time create a `longitude` column

Save each step as a JSON file

1) Undo/Redo

Click on the **Undo/ Redo** icon next to **Facet / Filter**

2) Six steps produce this data set

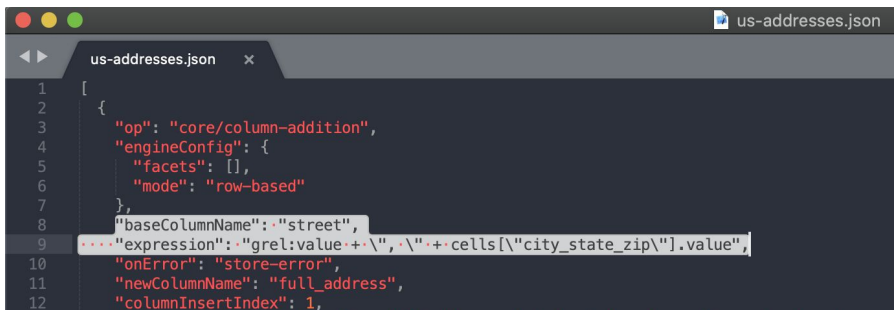Identify the steps needed to create this data set (6 of them)

Save each step as a JSON file

3) click on **Extract**, and copy + paste
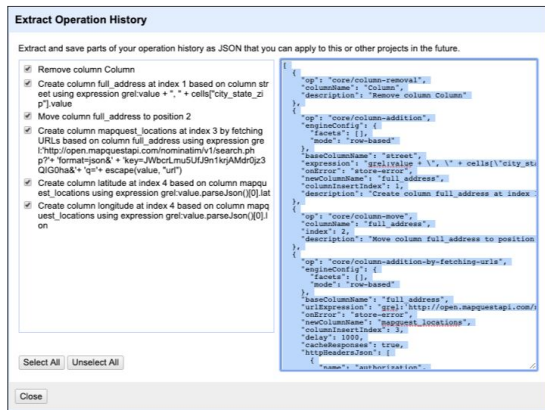
Click on the **Extract** button (next to **Apply**)

Make sure all the check the boxes!

- Copy + Paste this into a file and save for the next project!

Check out these links to learn more!

1. John R Little has a great <u>Pragmatic Datafication workshop</u> with some great materials (<u>slides</u> and <u>workbook</u>)

2. Check out the OpenRefine [GREL documentation on Github](#) (lots of great tidbits in here!)

3. Good 'ol fashioned YouTube! Check out the following video to learn more:

- [Data Journalism - Cleaning Data in Workbench and OpenRefine](#)

- [OpenRefine Beginners Tutorial](#)

- [Clean Your Data: Getting Started with OpenRefine](#)