Gro Intelligence Python API Client Introduction

Gro Intelligence makes its platform available to certain clients through its Python API Client. The API Client allows users to access all of the data through intuitive Python function calls, and returns the data in various pandas dataframe time series, or as a standard Python list of data points.

The first thing most users will need to do is to get authentication with the get_access_token() function. Gro Intelligence will give approved users an api_host, gro_user_email, and gro_user_password so you can use get_access_token():

```
access_token = api.client.lib.get_access_token( API_HOST, gro_user_email, gro_user_password )
```

The next step is to create a client object with the token generated:

```
client = api.client.Client(API_HOST, access_token)
```

The convenience functions that we use are all methods of the client object we just created.

In order to get the correct data, the user must specify a number of variables in integer formats: metric_id, item_id, region_id, frequency_id, and source_id, with bilateral trade series also needing a partner_region_id to be fully specified. Here are the Gro definitions of those terms:

Name	Definition	Example	Example ID
Item	Thing being measured	Soybeans	270
Metric	Attribute being measured	Export Volume (mass)	20032
Region	Location	Brazil	1029
Frequency	How often	Annual	9
Source	Who is the data provider	USDA PS&D	14
Partner Region	For bilateral data, who is the partner	China	1231

Given those five or six integers, the API will return usable data ready for further analysis.

The Gro API has a number of helper functions which help the user to determine the appropriate ID codes to use (entity_type is one of the following: 'items', 'metrics', 'regions', 'sources', 'units', 'frequencies'):

- search(entity_type, search_terms) - for when you have some words and you don't know what the integer id is - returns an integer
- lookup(entity_type, entity_id) - for when you have an id number but you don't know what the corresponding name is - returns a string
- translate(unknown_value) - for when you don't know what type of entity your unknown value might be (can be integer or string) and just want a list of possibilities

Let's use these functions to figure out how to get a series of interest:

- client.search('items', 'Soybeans')

The above call (currently) returns a list of 27 possibilities, which all include soybeans in their name but vary in sometimes-subtle ways. Here's the first row of the response, which is also the closest to the string we asked for:

```
{u'count': 109941,
u'definition': None,
u'id': 270,
u'name': u'Soybeans'}
```

So, the item_id we want is 270. Now we repeat the process for each of the values we need for our query:

- client.search('metrics', 'Production')
- client.search('regions', 'Brazil')
- client.search('frequencies', 'Annual')
- client.search('sources', 'USDA')

Now we have a complete set of integer id values which will be sufficient to define an annual (9) time series of USDA's(14) estimate of soybean(270) production(860032) in Brazil(1029).

Example - Getting Data

Now that we have the five or six integer entity_ids that we need to get data, it's a fairly simple matter to get it.

Here's the function which gets the data:

- df = get_df(client, item_id = 270, metric_id = 860032, region_id=1029, frequency_id = 9, source_id = 14)

Which will return a pandas DataFrame - here are some example rows:

	available_date	end_date	frequency_id	frequency_name	input_unit_id	input_unit_scale	item_id	metric_id	region_id	reporting_date	sampling_type	start_date	value
0	2017-09-16T00:00:00.000Z	1978-12-31T00:00:00.000Z	9	annual	14	1000	270	860032	1029	None	total	1978-01-01T00:00:00.000Z	9541
1	2017-09-16T00:00:00.000Z	1979-12-31T00:00:00.000Z	9	annual	14	1000	270	860032	1029	None	total	1979-01-01T00:00:00.000Z	10240
2	2017-09-16T00:00:00.000Z	1980-12-31T00:00:00.000Z	9	annual	14	1000	270	860032	1029	None	total	1980-01-01T00:00:00.000Z	15156
3	2017-09-16T00:00:00.000Z	1981-12-31T00:00:00.000Z	9	annual	14	1000	270	860032	1029	None	total	1981-01-01T00:00:00.000Z	15200

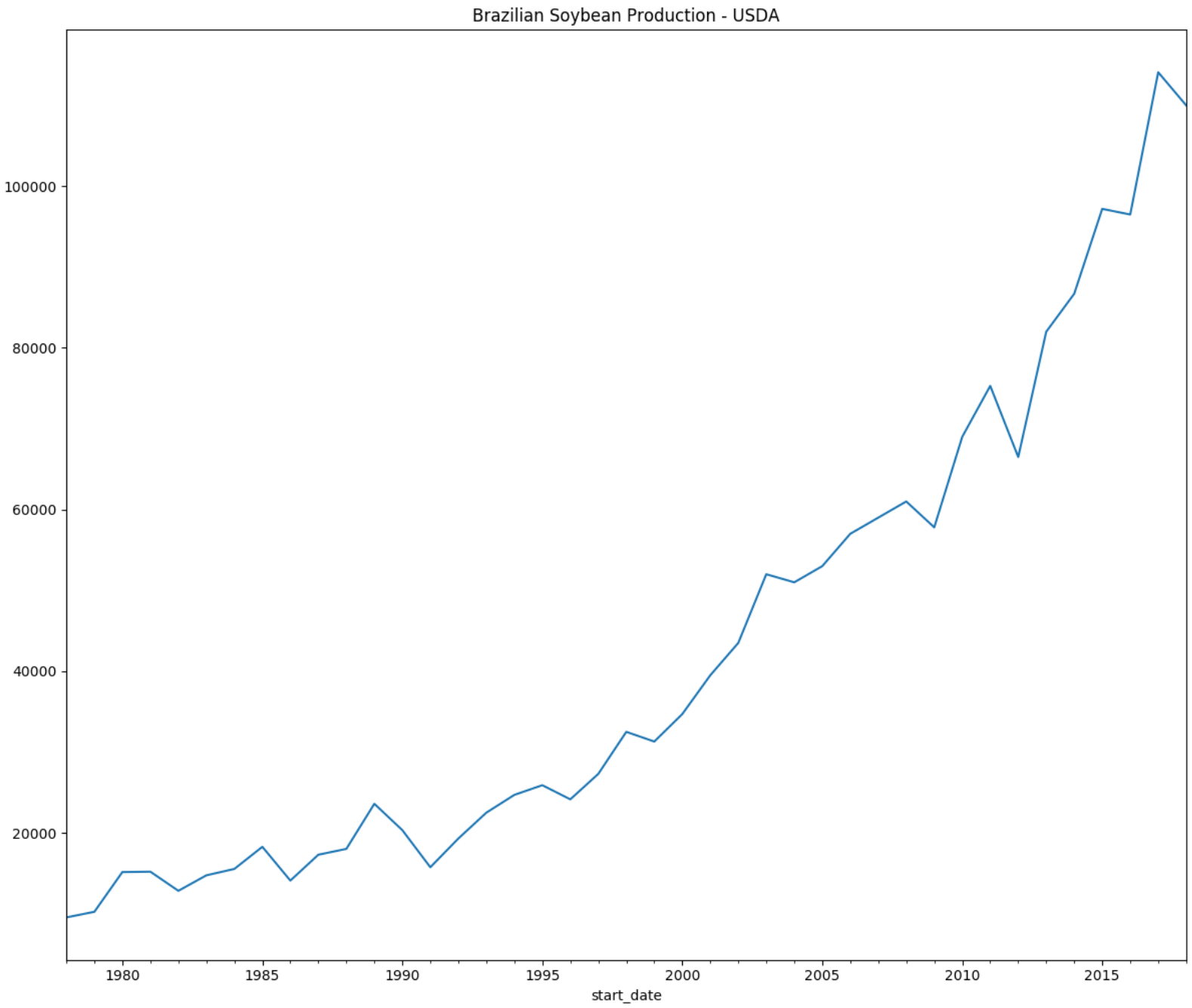
We should set the index to the start_date field for display:

```
df = df.set_index( pd.to_datetime( df['start_date'] ) )     #Format and set index
```

And then plot it:

```
df['value'].plot(title ='Brazilian Soybean Production - USDA')
```

Which produces this chart:



API Functions

Search through ontology

Search metrics: given keyword, all the metrics (id, name), where name or name of parent matches keyword

Search items: same as metrics

Search regions: same as metrics

Optionally filter by item, metric, region and partner region ids.

Examples:

https://api.gro-intelligence.com/v2/search/items?q=sesame

https://exp-api.gro-intelligence.com/v2/search/metrics?q=export+quantity

https://exp-api.gro-intelligence.com/v2/search/regions?q=nairobi

https://exp-api.gro-intelligence.com/v2/search/items?q=fruit&regionId=1215

Count available entities

Count # metrics, items, regions and partner regions available given metric, item, region and partner region filter...partner regions are only included in result if metric is selected.

Example:

https://exp-api.gro-intelligence.com/v2/entities/count?regionId=10000272&metricId=2310114&itemId=779

List available entities

List (metrics, items, regions) for which data series are available given metric, item, region, and partner region filters.
If metric has been selected, partner regions if any are also included in the results.

Example:

https://exp-api.gro-intelligence.com/v2/entities/list?regionId=1065&itemId=204&metricId=30032

Get data points

Given metric, item, region, partner region, frequency, source, return data values.

Examples:

https://exp-api.gro-intelligence.com/v2/data?regionId=1001&itemId=63&metricId=860032&frequency=annual&sourceId=2
https://exp-api.gro-intelligence.com/v2/data?sourceId=3&frequency=8-day&regionId=102014&metricId=70029&itemId=321

Lookup attribute by ID

Given an attribute (metric, item, region, partner region, frequency, or source), and in ID integer, return the meaning of that integer.

Examples:

client.lookup('items', 12)
client.lookup("units', 6)

Translate item

Given a string or an integer return the possible meanings in the Gro database (could generate a lot of replies).

Example:

client.translate('Brazil')
client.translate(12)

Get pandas DataFrame

Given a full list of ids, return a pandas dataframe.

Example:

get_df(client, item_id = 270, metric_id = 860032, region_id=1029, frequency_id = 9, source_id = 14)

Visit **www.gro-intelligence.com** for more information.

Email **info@gro-intelligence.com** to schedule a demo.

NEW YORK OFFICE

12 E 49th Street, 11th Floor
New York, NY, USA

+1 718 935 0100

NAIROBI OFFICE

Eldama Park
Tsavo Block, 4th Floor
Eldama Ravine Road
Westlands, Nairobi, Kenya

+254 788 448 678
+254 202 186 996