odbrasil Documentation

Release 0.2dev

Christian S. Perone

August 14, 2012

CONTENTS

Release v0.2dev. (*Installation*)

odbrasil is an *Apache 2.0 licensed* Python module to extract Brazilian government open data. The aim of the project is to provide an unified, organized and well-documented API to extract and parse (typically into Pandas data structures) the government open data.

Today we have some projects doing scraping of the open data, but these projects doesn't offer a parse for Pandas and do not have an unified and organized API, most of them are just *scripts* created in a hurry on Hackatons and do not have any documentation.

The API we're working on is simple and easy-to-use, intended not only for programmers but also for statisticians that doesn't have a strong background development.

We **really need** the community support in order to cover a great part of the API available for the government open data, if you want to help, join us on Github.

I have chosen the Pandas because it is becoming the *lingua franca* of the Python data analysis toolkits and because it is integrated with matplotlib and scipy/numpy ecosystem.

CHAPTER

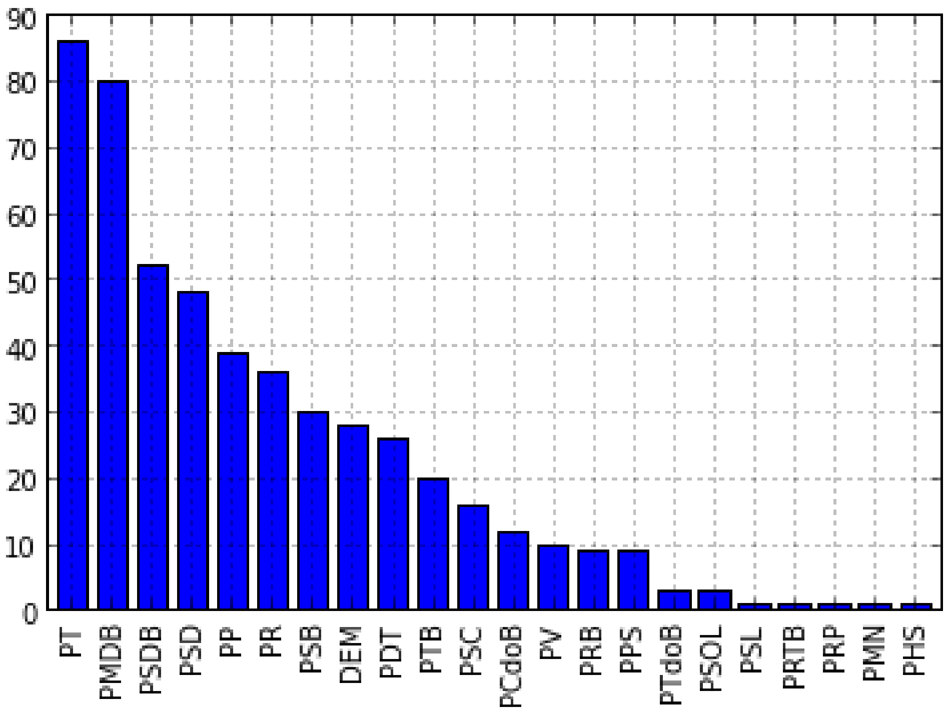
ONE

USAGE EXAMPLE: LEGISLATIVO.DEPUTADOS

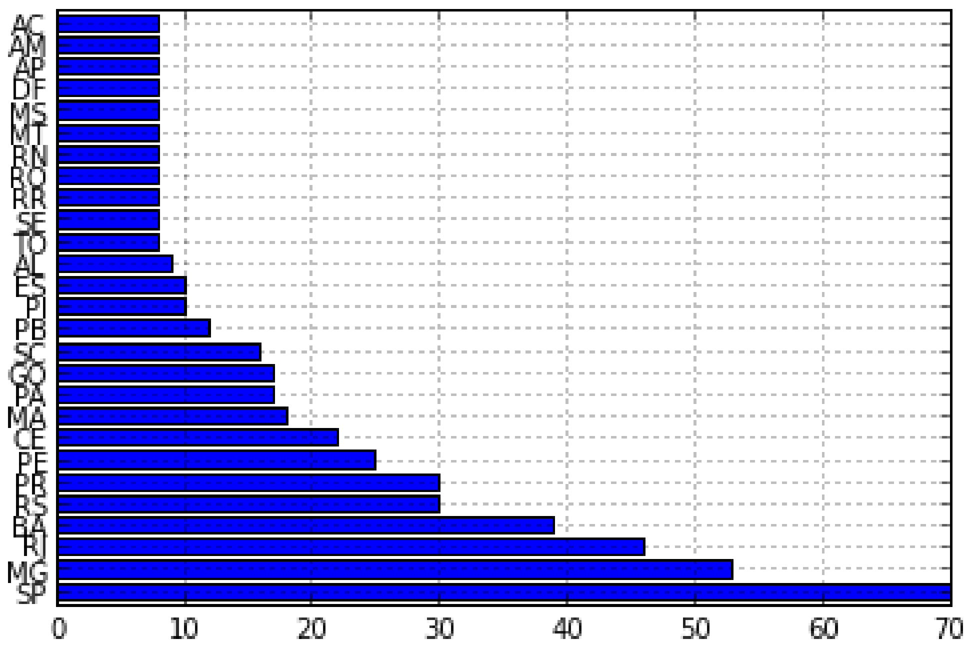
Here is an example of what the API can do:

```
>>> from odbrasil.legislativo import camara
>>> api = camara.Deputados()
>>> deputados = api.get_deputados()
>>> deputados
<class 'pandas.core.frame.DataFrame'>
Int64Index: 512 entries, 0 to 511
Data columns:
anexo               512  non-null values
email               512  non-null values
fone                512  non-null values
gabinete            512  non-null values
idParlamentar       512  non-null values
nome                512  non-null values
nomeParlamentar     512  non-null values
partido             512  non-null values
sexo                512  non-null values
uf                  512  non-null values
dtypes: object(10)

>>> vcounts = deputados.partido.value_counts()
>>> vcounts
PT        86
PMDB      80
PSDB      52
PSD       48
PP        39
PR        36
PSB       30
DEM       28
PDT       26
PTB       20
PSC       16
PCdoB     12
PV        10
PRB        9
PPS        9
PTdoB      3
PSOL       3
PSL        1
```

```
PRTB       1
PRP        1
PMN        1
PHS        1
>>> vcounts.plot(kind='bar')
```



```
>>> uf_deputados = deputados.uf.value_counts()
>>> uf_deputados.plot(kind='barh')
```



CHAPTER

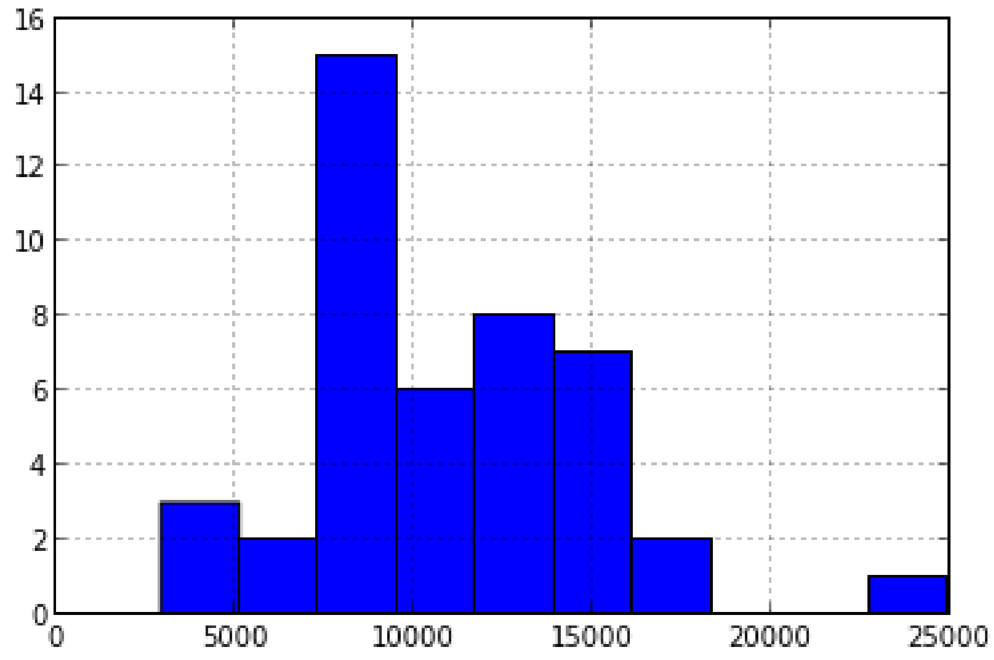
TWO

USAGE EXAMPLE: MONTHLY PAYMENTS OF UFRGS TEACHERS

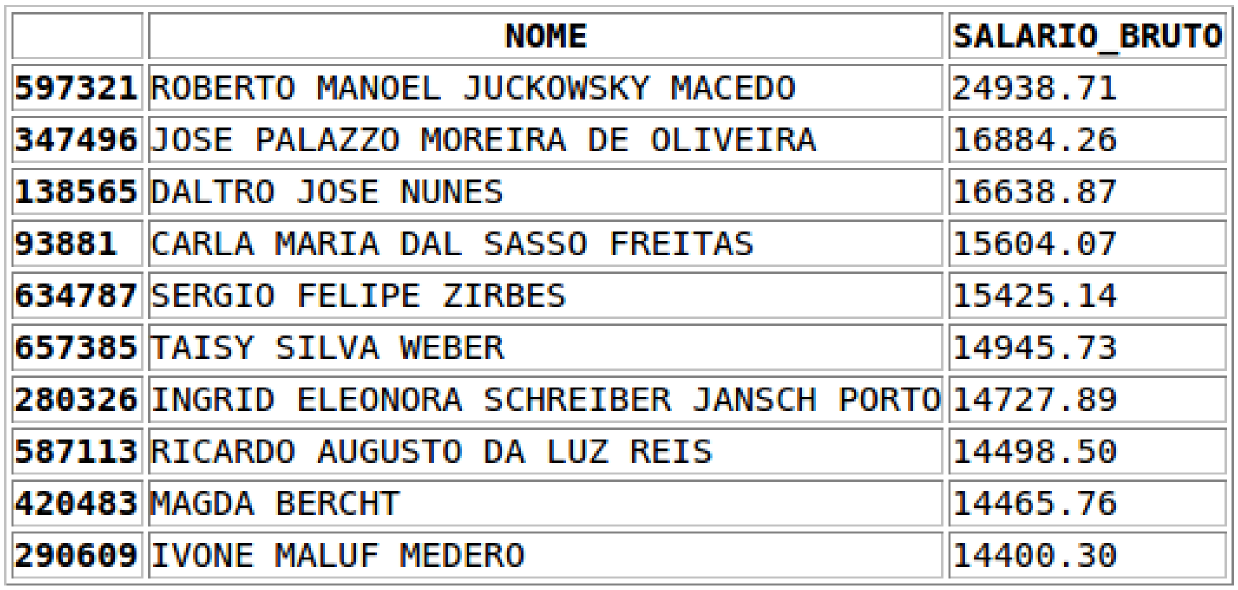
First, you have to download the CSV data called ‘Servidores 2009-2012’ from the government site, unfortunately this data has 44MB compressed and there is no REST API for that, so you have to download it because it cannot be shipped together with **odbrasil** package due to its size.

```
>>> import pandas
>>> from odbrasil.servidores import scrap
# This operation may take a while, it’s big file to parse
>>> dframe = pandas.read_csv(’servidores.csv’, sep=";")
>>> len(dframe)
706755
# We have now a DataFrame with 706k rows !

# I’m going to filter only UFRGS employees
>>> ufrgs_lotacao = ’UNIVERSIDADE FED. DO RIO GRANDE DO SUL’
>>> only_ufrgs = dframe[dframe.ORG_LOTACAO==ufrgs_lotacao]
>>> len(only_ufrgs)
6080

# It’s better now, but let’s filter only teachers
>>> professor_cargo = ’PROFESSOR 3 GRAU’
>>> teachers = only_ufrgs[only_ufrgs.DESCRICAO_CARGO==professor_cargo]
>>> len(teachers)
2428

# Let’s filter now only teachers from computer science department
>>> dep_informatica = ’DEPARTAMENTO DE INFORMATICA APLICADA’
>>> informatica = teachers[teachers.UORG_LOTACAO==dep_informatica]
>>> len(informatica)
48

# Now, let’s use the odbrasil scrap functions from the Servidores module
# to get the monthly payments of these teachers
# The get_salario_bruto will use the name of the teacher to get his
# monthly payment
>>> def get_salario_bruto(nome):
... try:
...     servidor_id = scrap.get_servidor_id(nome)
...     salario = scrap.get_servidor_remuneracao_bruta(servidor_id)
...     return salario
... except:
...     return np.nan
```

```
# And now we create a new column on the DataFrame with the new values
>>> informatica["SALARIO_BRUTO"] = informatica["NOME"].map(get_salario_bruto)

# You're free now to do data analysis using this new column data
>>> informatica["SALARIO_BRUTO"].describe()
count        44.000000
mean      10683.359091
std        4192.178186
min        2910.380000
25%        8023.490000
50%       11131.690000
75%       13381.347500
max       24938.710000

>>> informatica["SALARIO_BRUTO"].hist()
```

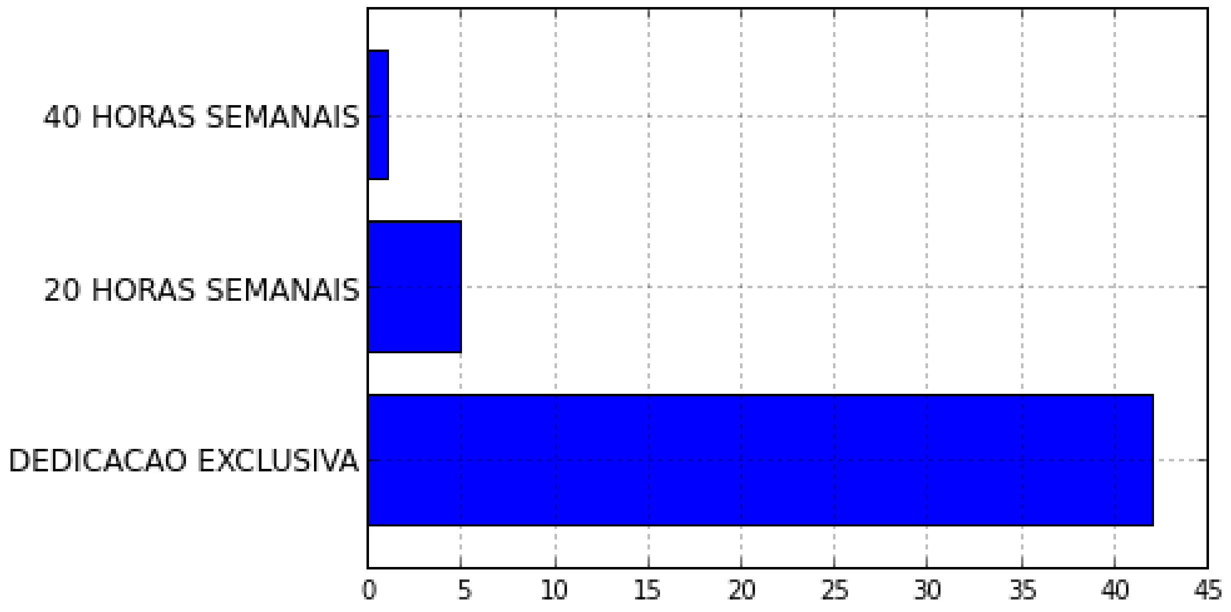


```
>>>
# Create a new DataFrame only with 2 columns: NOME, SALARIO_BRUTO
>>> nome_salario = informatica[["NOME", "SALARIO_BRUTO"]]

# Settings the NaNs to zero
>>> nome_salario["SALARIO_BRUTO"] = nome_salario["SALARIO_BRUTO"].fillna(0)

# Print the top-10
>>> nome_salario.sort_index(by='SALARIO_BRUTO', ascending=False)[0:10]
```



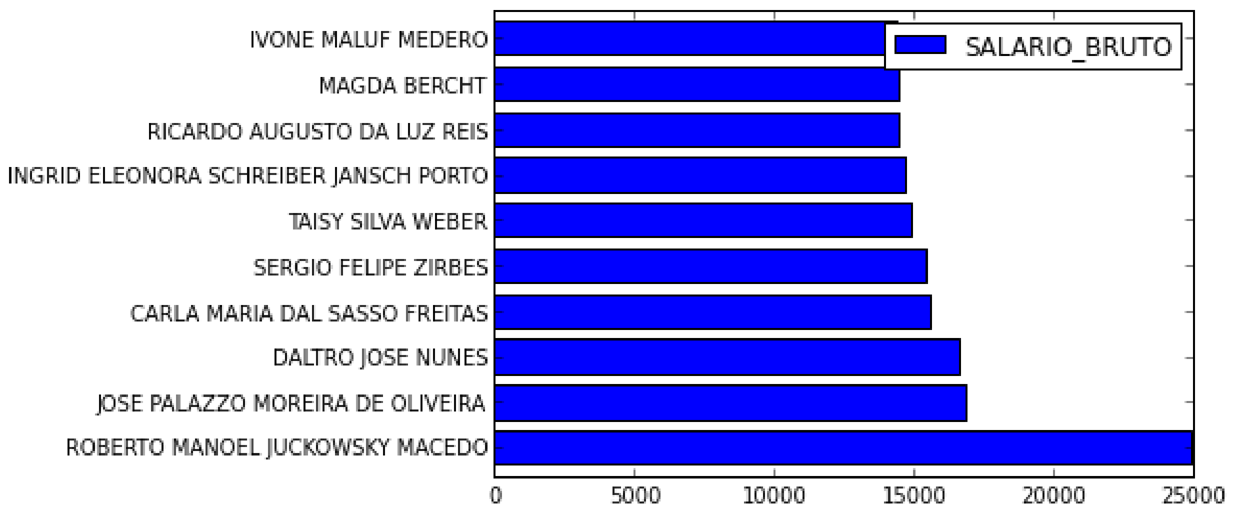
	NOME	SALARIO_BRUTO
597321	ROBERTO MANOEL JUCKOWSKY MACEDO	24938.71
347496	JOSE PALAZZO MOREIRA DE OLIVEIRA	16884.26
138565	DALTRO JOSE NUNES	16638.87
93881	CARLA MARIA DAL SASSO FREITAS	15604.07
634787	SERGIO FELIPE ZIRBES	15425.14
657385	TAISY SILVA WEBER	14945.73
280326	INGRID ELEONORA SCHREIBER JANSCH PORTO	14727.89
587113	RICARDO AUGUSTO DA LUZ REIS	14498.50
420483	MAGDA BERCHT	14465.76
290609	IVONE MALUF MEDERO	14400.30

```
>>>
# You can also do things like this:
>>> informatica.JORNADA_DE_TRABALHO.value_counts().plot(kind='barh')
```



::

```
>>> nome_salario.index = nome_salario["NOME"].values
>>> nome_salario.pop("NOME")
>>> nome_salario.sort('SALARIO_BRUTO', ascending=False)[0:10].plot(kind='barh')
```



And that's it, pretty easy don't you think ? See the API documentation and the Pandas documentation for more information.

CHAPTER

THREE

INSTALLATION

You can use **pip** to install **odbrasil** module and its dependencies, it is recommended that you have already installed scipy/numpy and matplotlib from your distro, in Ubuntu for instance:

```
sudo apt-get install python-numpy python-scipy python-matplotlib
```

And to install **odbrasil**:

```
pip install odbrasil
```

Simple and easy as that.

CHAPTER

FOUR

API DOCUMENTATION

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

4.1 API

This part of the documentation covers all the interfaces of **odbrasil**.

Note: **odbrasil** usually returns the data extracted as a Pandas DataFrame, to get a better introduction on how to use it, see Pandas documentation.

4.1.1 Legislativo

odbrasil.legislativo.camara

This module implements the methods to extract the information present on on the government site for open data: http://www2.camara.gov.br/transparencia/dados-abertos

class `odbrasil.legislativo.camara.`**`Deputados`**

Bases: `odbrasil.legislativo.camara.RESTServiceClient`

This class is responsible by accessing, extracting and parsing the data from the Deputados government endpoint.

`get_deputados`(*format='pandas'*, ***params*)

This method will get a Deputados list in various formats, use the `format` parameter to define which parameter you want to parse the data.

Parameters

- **format** – "pandas" or "xml"
- **params** – extra parameters will be redirected to Requests

Return type the parsed xml or the pandas *DataFrame*.

class `odbrasil.legislativo.camara.`**`Orgaos`**

Bases: `odbrasil.legislativo.camara.RESTServiceClient`

This class is responsible by accessing, extracting and parsing the data from the Orgaos government endpoint.

`get_orgaos` (*format='pandas'*, ***params*)

This method will get a Orgaos list in various formats, use the `format` parameter to define which parameter you want to parse the data.

Parameters

- **format** – "pandas" or "xml"
- **params** – extra parameters will be redirected to Requests

Return type the parsed xml or the pandas *DataFrame*.

`get_tipos_orgao` (*format='pandas'*, ***params*)

This method will get a list of 'Tipos de Orgaos'. Use the `format` parameter to define which parameter you want to parse the data.

Parameters

- **format** – "pandas" or "xml"
- **params** – extra parameters will be redirected to Requests

Return type the parsed xml or the pandas *DataFrame*.

class `odbrasil.legislativo.camara.`**`RESTServiceClient`**

Bases: `object`

The base class used by other subclasses to retrieve data from the government webservices. If you want to subclass this class, you have to define two class variables on your subclass, called `base_url` and the expected `content_type`. See `Deputados` for reference.

This class is responsible for keeping the common functionality used by the service clients, like using the **User-Agent** as **odbrasil/1.0** for instance.

Note: you shouldn't use this `RESTServiceClient` on your application except if you really need to customize the internals of the REST client.

`get` (*service*, ***params*)

This method uses the `baseurl` parameter and concats the `service` parameter into it to create the request URL. Any extra param passed to this method by the `params` parameter will be redirected to the Requests request.

Parameters

- **service** – the service, i.e. 'ObterDeputados'
- **params** – extra parameters to be used by Requests

Return type the Requests request response

`odbrasil.legislativo.camara.`**`pandas_parse_deputados`** (*xml_deputado_list*)

Method used to parse a xml parsed list of `deputado` elements into a pandas *DataFrame*.

Parameters **xml_deputado_list** – the xml parsed data returned by calling `Deputados.get_deputados()` with the `format` as 'xml' instead of 'pandas'.

Return type pandas 'DataFrame'

odbrasil.legislativo.camara.**pandas_parse_only_attributes**(*xml_list*)

This method converts a list of xml elements containing only attributes (without childs) to a pandas *DatFrame*.

Parameters **xml_list** –

Return type pandas *DataFrame*

4.1.2 Servidores

odbrasil.servidores.scrap

This module implements the methods to extract the information present on on the government site for open data: http://www.portaldatransparencia.gov.br/servidores/index.asp

Note: The methods present in this module are intended to do web scraping of the data from the "Portal da Transparencia", since the government doesn't provide n decent REST API for that service yet.

odbrasil.servidores.scrap.**get_servidor_id**(*name*)

This function will load the `IdServidor` parameter from the government site, it will use scraping methods and not a REST API to do that. The `name` parameter must be unique, otherwise an exception will be raised.

Parameters **name** – The servidor name

Return type The id of the servidor on the government database

odbrasil.servidores.scrap.**get_servidor_remuneracao_bruta**(*servidor_id*)

This function will load the month payment for the 'Servidor', it will use scraping methods and not a REST API to do that.

Parameters **servidor_id** – The servidor id, returned by `get_servidor_id()`

Return type The month payment of that 'Servidor'

CHAPTER

FIVE

LICENSE

5.1 Contributors

Christian S. Perone [twitter] [blog] [github].

PYTHON MODULE INDEX