

Notebook

September 15, 2014

1 Connecting with Wikimedia API

1.1 Raw connection

It is possible to connect directly to the Wikimedia API. In order to do so, a URL must be constructed specifically to query for the required data. The Wikimedia API will then respond on the requested format (JSON in this case, but could be XML or other supported formats). The Wikimedia API is described [here](#). The API [sandbox](#) can be used to construct URL queries for different kinds of data.

NOTE: The API can return a maximum of 500 results per request. If the performed query contains more than 500 items, the first 500 along with a continuation id. A subsequent request must be issued to retrieve the next 500 items, until no continuation token is present.

```
In [1]: %pylab inline
```

Populating the interactive namespace from numpy and matplotlib

```
In [2]: import urllib
import json

import wikidat.utils.ipresolver as resolver
import btb.utils.tools as btbtb
from __future__ import division
```

1.1.1 Query revisions for a given page

We start by building an API request to retrieve information from a given page title. In this case we are interested in revision information for the requested page, and more specifically, on the user information for each revision. This means for a given page title (e.g. ‘Amsterdam’) we will get a list of all users which have made a contribution to this page. This [URL](#) is an example request, and it shows the JSON response we will get back.

The response contains a list named “revisions”, whose elements represent the user information for each revision. Each revision element contains a “user” element, which is either the username OR the user IP. The user IP is returned when the revision was made by an anonymous user. These revision elements are identifiable because they also contain a “anon” element, which marks them as an anonymous revision.

```
In [3]: def nextRevisionRequest(title, rvcontinue=None):
        """
        Perform a wikipedia API revision request. This request returns
        the JSON structure of the response.

        title          Title of the requested wikipedia page
        rvcontinue      Continuation token (if known) (default=None). Used to
                        continue retrieving a previously started query, and
                        revision list starts from the given point. If None,
```



```

{u'user': u'JamesBWatson'},
{u'user': u'Nslospeed'},
{u'user': u'Tomgreep'},
{u'anon': u'', u'user': u'173.77.169.32'},
{u'user': u'AnomieBOT'},
{u'anon': u'', u'user': u'217.120.225.80'},
{u'anon': u'', u'user': u'217.120.225.80'},
{u'anon': u'', u'user': u'217.120.225.80'},
{u'user': u'Gemini1980'},
{u'anon': u'', u'user': u'91.109.6.226'},
{u'anon': u'', u'user': u'91.109.6.226'},
{u'user': u'Brandmeister'},
{u'user': u'Brandmeister'},
{u'user': u'Brandmeister'},
{u'user': u'Brandmeister'},
{u'user': u'Waldo79'},
{u'user': u'Thayts'},
{u'user': u'Thayts'},
{u'user': u'Thayts'},
{u'user': u'Thayts'},
{u'anon': u'', u'user': u'212.69.46.83'},
{u'user': u'Matthew M Kerr'},
{u'anon': u'', u'user': u'62.194.150.95'},
{u'user': u'J 1982'},
{u'user': u'Alensmitz'},
{u'anon': u'', u'user': u'179.236.98.179'},
{u'user': u'Gati123'},
{u'user': u'The Almighty Drill'},
{u'user': u'The Almighty Drill'},
{u'user': u'BattyBot'},
{u'user': u'Boerenkaasje'},
{u'anon': u'', u'user': u'182.64.209.89'},
{u'anon': u'', u'user': u'5.79.72.193'},
{u'anon': u'', u'user': u'95.146.148.30'},
{u'anon': u'', u'user': u'95.146.148.30'},
{u'anon': u'', u'user': u'95.146.148.30'},
{u'user': u'Flyer22'},
{u'user': u'LIAMBY49'},
{u'anon': u'', u'user': u'2.125.161.93'},
{u'user': u'Janwillemvanaalst'},
{u'anon': u'', u'user': u'122.168.228.84'},
{u'anon': u'', u'user': u'122.168.228.84'},
{u'user': u'JL-Bot'},
{u'user': u'AnomieBOT'},
{u'user': u'Adam Cuerden'},
{u'user': u'Javiereq'},
{u'user': u'Amortias'},
{u'anon': u'', u'user': u'207.250.52.225'},
{u'user': u'Abcjake'},
{u'anon': u'', u'user': u'187.208.124.191'},
{u'user': u'Jermerc'},
{u'user': u'Jermerc'},
{u'user': u'Marek69'},
{u'user': u'D Eaketts'},

```

```

{u'anon': u'', u'user': u'213.46.27.233'},
{u'user': u'Ben.michel'},
{u'user': u'Blue520'},
{u'anon': u'', u'user': u'213.46.27.233'},
{u'user': u'Magioladitis'},
{u'anon': u'', u'user': u'68.82.15.227'},
{u'user': u'Marcel.Vermeer'},
{u'user': u'Marcel.Vermeer'},
{u'user': u'Sangam'},
{u'user': u'Magioladitis'},
{u'user': u'Allardo'},
{u'user': u'Allardo'},
{u'user': u'Allardo'},
{u'user': u'Allardo'},
{u'anon': u'', u'user': u'78.27.63.163'},
{u'user': u'Nslospeed'},
{u'user': u'Nslospeed'},
{u'user': u'K6ka'},
{u'user': u'Chris0693'},
{u'user': u'Allardo'},
{u'user': u'Allardo'},
{u'user': u'Ben.michel'},
{u'anon': u'', u'user': u'83.80.254.21'},
{u'user': u'Sangam'},
{u'anon': u'', u'user': u'83.100.211.100'},
{u'anon': u'', u'user': u'124.168.165.81'},
{u'user': u'Zzyzx11'},
{u'user': u'Edit\x8r'},
{u'user': u'Edit\x8r'},
{u'user': u'Monkbot'},
{u'anon': u'', u'user': u'80.126.166.218'},
{u'anon': u'', u'user': u'80.126.166.218'},
{u'anon': u'', u'user': u'46.116.154.76'},
{u'user': u'Iltever'},
{u'user': u'M-le-mot-dit'},
{u'anon': u'', u'user': u'178.134.148.242'},
{u'user': u'Japanese Rail Fan'},
{u'user': u'Wavelength'},
{u'user': u'RjwilmsiBot'},
{u'anon': u'', u'user': u'81.88.222.36'},
{u'user': u'JamesBWatson'},
{u'anon': u'', u'user': u'75.191.173.190'},
{u'user': u'AnomieBOT'},
{u'user': u'JamesRussels'},
{u'user': u'Tony Holkham'},
{u'anon': u'', u'user': u'85.179.64.179'},
{u'user': u'IcheckyoursPELL'},
{u'anon': u'', u'user': u'83.160.84.254'},
{u'user': u'Trappist the monk'},
{u'user': u'Monkbot'},
{u'user': u'IronGargoyle'},
{u'anon': u'', u'user': u'74.77.24.68'},
{u'user': u'Luxorr'},
{u'user': u'ClueBot NG'},

```

```

{u'anon': u'', u'user': u'99.227.137.88'},
{u'user': u'Hazelares'},
{u'user': u'Friezer'},
{u'user': u'Janwillemvanaalst'},
{u'user': u'Eyesnore'},
{u'anon': u'', u'user': u'187.78.26.184'},
{u'user': u'ClueBot NG'},
{u'anon': u'', u'user': u'187.78.26.184'},
{u'user': u'Hazelares'},
{u'user': u'Hazelares'},
{u'user': u'Hazelares'},
{u'user': u'Hazelares'},
{u'anon': u'', u'user': u'37.74.40.2'},
{u'anon': u'', u'user': u'82.139.77.229'},
{u'user': u'ClueBot NG'},
{u'anon': u'', u'user': u'2.83.219.7'},
{u'user': u'CRwikiCA'},
{u'anon': u'', u'user': u'81.88.222.36'},
{u'user': u'Tucoxn'},
{u'anon': u'', u'user': u'86.81.207.142'},
{u'user': u'Diannaa'},
{u'user': u'Buxtehude'},
{u'user': u'Bgwhite'},
{u'user': u'GuppieB52'},
{u'user': u'Buxtehude'},
{u'user': u'Buxtehude'},
{u'user': u'CRwikiCA'},
{u'user': u'\u0639\u0645\u064a\u062f \u0637\u0627\u0647\u0631'},
{u'user': u'Chris the speller'},
{u'user': u'Chris the speller'},
{u'anon': u'', u'user': u'70.114.162.81'},
{u'anon': u'', u'user': u'70.114.162.81'},
{u'user': u'Josh3580'},
{u'anon': u'', u'user': u'103.14.60.43'},
{u'anon': u'', u'user': u'103.14.60.43'},
{u'user': u'Jared Preston'},
{u'anon': u'', u'user': u'91.40.125.175'},
{u'user': u'Jon335'},
{u'user': u'Widr'},
{u'anon': u'', u'user': u'173.72.38.106'},
{u'user': u'Bender235'},
{u'user': u'Bender235'},
{u'anon': u'', u'user': u'62.140.132.24'},
{u'anon': u'', u'user': u'62.140.132.24'},
{u'anon': u'', u'user': u'62.140.132.24'},
{u'user': u'Wavelength'},
{u'user': u'K6ka'},
{u'anon': u'', u'user': u'24.1.195.163'},
{u'user': u'VKing'},
{u'user': u'Stephaela'},
{u'user': u'VKing'},
{u'user': u'VKing'},
{u'user': u'VKing'},
{u'user': u'VKing'},

```

```

{u'user': u'BG19bot'},
{u'user': u'VKing'},
{u'user': u'VKing'},
{u'user': u'VKing'},
{u'anon': u'', u'user': u'129.252.94.222'},
{u'anon': u'', u'user': u'217.122.213.14'},
{u'user': u'Monkbot'},
{u'user': u'Gilliam'},
{u'anon': u'', u'user': u'86.82.143.237'},
{u'anon': u'', u'user': u'86.82.143.237'},
{u'user': u'BattyBot'},
{u'anon': u'', u'user': u'192.87.10.88'},
{u'user': u'Kykams'},
{u'user': u'Kykams'},
{u'anon': u'', u'user': u'72.187.40.190'},
{u'user': u'A13ean'},
{u'user': u'Tezcan Varol'},
{u'user': u'ChrisGualtieri'},
{u'user': u'Mmortal03'},
{u'anon': u'', u'user': u'69.1.34.84'},
{u'user': u'Materialscientist'},
{u'anon': u'', u'user': u'194.120.70.122'},
{u'user': u'Marek69'},
{u'anon': u'', u'user': u'80.254.146.156'},
{u'anon': u'', u'user': u'80.254.146.156'},
{u'anon': u'', u'user': u'198.162.93.5'},
{u'user': u'Chris0693'},
{u'user': u'Chris0693'},
{u'anon': u'', u'user': u'5.150.92.20'},
{u'user': u'Ricqk'},
{u'user': u'I am One of Many'},
{u'anon': u'', u'user': u'146.90.254.179'},
{u'anon': u'', u'user': u'82.217.85.47'},
{u'anon': u'', u'user': u'82.217.85.47'},
{u'anon': u'', u'user': u'82.217.85.47'},
{u'user': u'HMSSolent'},
{u'anon': u'', u'user': u'86.138.55.217'},
{u'user': u'Foofik'},
{u'user': u'Philip Trueman'},
{u'anon': u'', u'user': u'64.82.207.134'},
{u'user': u'ClueBot NG'},
{u'anon': u'', u'user': u'50.200.28.42'},
{u'user': u'K6ka'},
{u'anon': u'', u'user': u'192.107.142.1'},
{u'user': u'M2545'},
{u'anon': u'', u'user': u'82.157.42.118'},
{u'anon': u'', u'user': u'86.134.209.31'},
{u'user': u'Some jerk on the Internet'},
{u'anon': u'', u'user': u'173.15.177.209'},
{u'user': u'Donner60'},
{u'anon': u'', u'user': u'173.15.177.209'},
{u'user': u'Gidonb'},
{u'user': u'Marek69'},
{u'anon': u'', u'user': u'82.157.42.118'},

```

```

{u'user': u'Velocitas'},
{u'user': u'MrTree'},
{u'anon': u'', u'user': u'213.124.217.164'},
{u'user': u'Reverend Mick man34'},
{u'user': u'Countrymaster'},
{u'user': u'Countrymaster'},
{u'user': u'Reverend Mick man34'},
{u'user': u'Reverend Mick man34'},
{u'user': u'Countrymaster'},
{u'user': u'MrTree'},
{u'user': u'Reverend Mick man34'},
{u'user': u'Widr'},
{u'anon': u'', u'user': u'86.86.184.77'},
{u'user': u'Sander.v.Ginkel'},
{u'user': u'M48b'},
{u'anon': u'', u'user': u'205.153.89.193'},
{u'anon': u'', u'user': u'131.211.215.126'},
{u'anon': u'', u'user': u'131.211.215.126'},
{u'anon': u'', u'user': u'131.211.215.126'},
{u'anon': u'', u'user': u'131.211.215.126'},
{u'anon': u'', u'user': u'131.211.215.126'},
{u'anon': u'', u'user': u'131.211.215.126'},
{u'user': u'Lugia2453'},
{u'anon': u'', u'user': u'69.91.185.216'},
{u'user': u'Lugia2453'},
{u'anon': u'', u'user': u'69.91.185.216'},
{u'user': u'ClueBot NG'},
{u'anon': u'', u'user': u'46.1.57.21'},
{u'user': u'Hinrik'},
{u'user': u'Fitnr'},
{u'anon': u'', u'user': u'83.160.84.254'},
{u'anon': u'', u'user': u'83.160.84.254'},
{u'user': u'Dutchlad1985'},
{u'user': u'Moswento'},
{u'user': u'Aisteco'},
{u'user': u'Atethnekos'},
{u'anon': u'', u'user': u'71.228.148.125'},
{u'anon': u'', u'user': u'92.108.122.122'},
{u'anon': u'', u'user': u'92.108.122.122'},
{u'anon': u'', u'user': u'92.108.122.122'},
{u'anon': u'', u'user': u'92.108.122.122'},
{u'user': u'Mahmudmasri'},
{u'anon': u'', u'user': u'77.170.53.91'},
{u'anon': u'', u'user': u'80.101.6.75'},
{u'anon': u'', u'user': u'178.85.254.20'},
{u'user': u'SiBr4'},
{u'anon': u'', u'user': u'87.232.47.10'},
{u'user': u'AlexanderVanLoon'},
{u'user': u'AlexanderVanLoon'},
{u'user': u'Edit\x8r'},
{u'user': u'Edit\x8r'},
{u'user': u'CommonsDelinker'},
{u'user': u'Marek69'},
{u'user': u'Materialscientist'},
{u'anon': u'', u'user': u'87.232.47.10'},
{u'user': u'Neuroscientific'},

```

```

{u'user': u'Neuroscientific'},
{u'user': u'CommonsDelinker'},
{u'user': u'Marek69'},
{u'anon': u'', u'user': u'125.161.215.25'},
{u'user': u'Baswerkhoven'},
{u'user': u'Marek69'},
{u'user': u'Swimmerguy269'},
{u'user': u'Swimmerguy269'},
{u'user': u'SiBr4'},
{u'user': u'Martarius'},
{u'user': u'Cydebot'},
{u'anon': u'', u'user': u'84.107.247.212'},
{u'user': u'MrTree'},
{u'user': u'MrTree'},
{u'user': u'MrTree'},
{u'user': u'Joshua Doubek'},
{u'user': u'Joshua Doubek'},
{u'user': u'Joshua Doubek'},
{u'user': u'Yoman82'},
{u'anon': u'', u'user': u'62.195.118.86'},
{u'user': u'CRwikiCA'},
{u'user': u'Mrpepertje1'},
{u'user': u'Mrpepertje1'},
{u'user': u'\u05d9\u05d5\u05e0\u05d4 \u05d1\u05e0\u05d3\u05dc\u05d0\u05e7'},
{u'anon': u'', u'user': u'184.65.66.185'},
{u'user': u'Pieter Dijkstra'},
{u'user': u'Thayts'},
{u'user': u'Thayts'},
{u'user': u'Thayts'},
{u'user': u'Thayts'},
{u'anon': u'', u'user': u'83.160.84.254'},
{u'anon': u'', u'user': u'83.160.84.254'},
{u'anon': u'', u'user': u'77.170.119.192'},
{u'anon': u'', u'user': u'77.170.119.192'},
{u'user': u'Thayts'},
{u'user': u'Thayts'},
{u'anon': u'', u'user': u'145.221.52.104'},
{u'user': u'Crom1'},
{u'user': u'Djflem'},
{u'user': u'Thayts'},
{u'user': u'Thayts'},
{u'user': u'Thayts'},
{u'anon': u'', u'user': u'163.231.6.88'},
{u'user': u'Pinethicket'},
{u'anon': u'', u'user': u'77.73.8.71'},
{u'anon': u'', u'user': u'86.4.70.35'},
{u'anon': u'', u'user': u'86.4.70.35'},
{u'anon': u'', u'user': u'86.4.70.35'},
{u'anon': u'', u'user': u'86.4.70.35'},
{u'anon': u'', u'user': u'121.44.65.199'},
{u'user': u'Cydebot'},
{u'user': u'Oldies1360'},
{u'user': u'Donner60'},
{u'anon': u'', u'user': u'123.100.99.181'},

```



```

{u'anon': u'', u'user': u'82.74.143.46'},
{u'user': u'Drmies'},
{u'user': u'Mcewan'},
{u'anon': u'', u'user': u'194.187.32.80'},
{u'user': u'FrescoBot'},
{u'user': u'Khazar2'},
{u'user': u'Drmies'},
{u'user': u'Drmies'},
{u'user': u'Drmies'},
{u'user': u'Drmies'},
{u'user': u'Drmies'},
{u'user': u'Drmies'},
{u'user': u'Drmies'},
{u'user': u'Drmies'},
{u'user': u'Drmies'},
{u'user': u'ImageRemovalBot'},
{u'user': u'Thayts'},
{u'user': u'ClueBot NG'},
{u'anon': u'', u'user': u'164.92.250.25'},
{u'user': u'P199'},
{u'anon': u'', u'user': u'83.160.84.254'},
{u'anon': u'', u'user': u'83.160.84.254'},
{u'anon': u'', u'user': u'83.160.84.254'},
{u'user': u'Friezer'},
{u'anon': u'', u'user': u'184.57.20.200'},
{u'user': u'ChrisGualtieri'},
{u'anon': u'', u'user': u'108.254.160.23'},
{u'anon': u'', u'user': u'108.254.160.23'},
{u'anon': u'', u'user': u'108.254.160.23'},
{u'anon': u'', u'user': u'108.254.160.23'},
{u'anon': u'', u'user': u'108.254.160.23'},
{u'anon': u'', u'user': u'31.153.5.151'},
{u'user': u'Martijn Hoekstra'},
{u'anon': u'', u'user': u'70.56.113.163'},
{u'anon': u'', u'user': u'77.193.163.69'},
{u'anon': u'', u'user': u'82.26.206.126'},
{u'user': u'Yobot'},
{u'user': u'Denny'},
{u'user': u'CommonsDelinker'},
{u'anon': u'', u'user': u'83.160.84.254'},
{u'anon': u'', u'user': u'92.239.225.189'},
{u'anon': u'', u'user': u'83.160.84.254'},
{u'user': u'Oxfordwang'},
{u'anon': u'', u'user': u'165.138.225.1'},
{u'user': u'Escottf'},
{u'user': u'AnomieBOT'},
{u'user': u'Go slowly'},
{u'user': u'Marek69'},
{u'anon': u'', u'user': u'109.131.20.149'},
{u'user': u'Magioladitis'},
{u'user': u'AnomieBOT'},
{u'user': u'CRwikiCA'},
{u'user': u'Palnatoke'},
{u'user': u'O.Koslowski'},

```

```

    {u'anon': u'', u'user': u'88.115.19.84'},
    {u'user': u'SporkBot'},
    {u'user': u'Mcewan'},
    {u'anon': u'', u'user': u'83.81.202.159'},
    {u'anon': u'', u'user': u'83.81.202.159'},
    {u'user': u'Marek69'},
    {u'anon': u'', u'user': u'94.210.219.224'},
    {u'anon': u'', u'user': u'24.212.154.203'},
    {u'user': u'Closedmouth'},
    {u'anon': u'', u'user': u'193.1.187.82'},
    {u'user': u'CRwikiCA'},
    {u'user': u'Yobot'},
    {u'anon': u'', u'user': u'78.105.238.95'},
    {u'anon': u'', u'user': u'78.105.238.95'},
    {u'user': u'\u0412\u043b\u0430\u0434\u0438\u043c\u0438\u0440 \u0428\u0435\u043b\u0444\u043f'},
    {u'user': u'Swimmeguy269'},
    {u'user': u'Swimmeguy269'},
    {u'user': u'\u0412\u043b\u0430\u0434\u0438\u043c\u0438\u0440 \u0428\u0435\u043b\u0444\u043f'},
    {u'user': u'\u0412\u043b\u0430\u0434\u0438\u043c\u0438\u0440 \u0428\u0435\u043b\u0444\u043f'},
    {u'user': u'Vrenator'},
    {u'anon': u'', u'user': u'77.225.116.244'},
    {u'user': u'Doddmeister47'},
    {u'user': u'Doddmeister47'},
    {u'user': u'\u0412\u043b\u0430\u0434\u0438\u043c\u0438\u0440 \u0428\u0435\u043b\u0444\u043f'},
    {u'user': u'\u0412\u043b\u0430\u0434\u0438\u043c\u0438\u0440 \u0428\u0435\u043b\u0444\u043f'},
    {u'user': u'Marek69'},
    {u'anon': u'', u'user': u'92.157.93.36'},
    {u'user': u'ClueBot NG'},
    {u'anon': u'', u'user': u'108.162.133.92'},
    {u'user': u'\u0412\u043b\u0430\u0434\u0438\u043c\u0438\u0440 \u0428\u0435\u043b\u0444\u043f'},
    {u'user': u'\u0412\u043b\u0430\u0434\u0438\u043c\u0438\u0440 \u0428\u0435\u043b\u0444\u043f'},
    {u'user': u'K7L'},
    {u'anon': u'', u'user': u'213.46.102.18'},
    {u'user': u'Aperson100'},
    {u'anon': u'', u'user': u'83.81.144.129'},
    {u'user': u'Swimmeguy269'},
    {u'anon': u'', u'user': u'217.122.73.67'},
    {u'anon': u'', u'user': u'217.122.73.67'},
    {u'anon': u'', u'user': u'217.122.73.67'},
    {u'anon': u'', u'user': u'217.122.73.67'},
    {u'user': u'Marek69'},
    {u'user': u'Swimmeguy269'},
    {u'user': u'Carl James Elitz'}],
    u'title': u'Amsterdam'}}},
    u'query-continue': {u'revisions': {u'rvcontinue': 536942827}}}}

```

```

In [5]: def isAnon(rev):
    """
    Determine whether a given revision is anonymous or not.
    Anonymous revision elements MUST contain an 'anon' element.
    """
    return 'anon' in rev.keys()

```

As mentioned before, if a *rvcontinue* element is present on the response, more results are available and a subsequent query is required. The following function retrieves the full revision history for a given page:

```

In [6]: def getContributionsByIP(pageTitle):
        '''
        Retrieve revision history of a given page.

        Returns a list of IP's (anon revisions) which contributed to
        that page, and a total number of revisions (user revisions
        + anon revisions)
        '''
        nRevs = 0
        anonIPs = []

        rvcontinue = None
        keepGoing = True

        while keepGoing:
            resp = nextRevisionRequest(pageTitle, rvcontinue)

            if 'query-continue' in resp:
                rvcontinue = resp['query-continue']['revisions']['rvcontinue']
            else:
                keepGoing = False

            pageIds = resp['query']['pages'].keys()
            assert len(pageIds)==1
            pageId = pageIds[0]
            revs = resp['query']['pages'][pageId]['revisions']

            for rev in revs:
                nRevs += 1
                if isAnon(rev):
                    anonIPs.append(rev['user'])
        return anonIPs, nRevs

```

The following is an example of the IP's contributing to a single page.

NOTE: Notice that registered users do not provide any useful information (except their name) and thus are ignored

```

In [7]: pageTitle = 'Amsterdam'
        ips, nRevs = getContributionsByIP(pageTitle)

        print 'Revision history for page: ' + pageTitle
        print '  Number of total revisions      : {:,}'.format(nRevs)
        print '  Number of anonymous revisions: {:,}'.format(len(ips))
        print '  Contribution IP\'s (first 10) : '
        print '      ',ips[:10]

```

Revision history for page: Amsterdam

Number of total revisions : 7,227

Number of anonymous revisions: 2,771

Contribution IP's (first 10) :

[u'108.254.160.23', u'2001:610:1908:C000:480D:5D05:FBB7:4AD5', u'173.77.169.32', u'217.120.225.80

We can now use these IP's to identify the originating country of each revision. For this, we use the *wikidat.utils.ipresolver* module.

```
In [8]: def countCountryContributions(ips):
        '''
        Convert a list of IP's to a map of contributions by country code.

        ips      List of IP addresses

        Returns a list map of contributions for each country code.
        '''
        return btbtk.countContributions(ips, resolver.getCountryCode)
```

The following example shows the number of contributions from selected countries.

```
In [9]: contribs = countCountryContributions(ips)
        for cc in [ 'US', 'UK', 'NL', 'CA' ]:
            print 'Contributions from {:}: {:,}'.format(cc,contribs[cc])
```

```
Contributions from US: 992
Contributions from UK: 241
Contributions from NL: 883
Contributions from CA: 139
```

The number of contributions to a single page from individual countries can be visualized as a pie chart.

```
In [10]: def displayResults(contribs, nRevs, pageTitle):
        '''
        Create pie chart visualization of country code contributions.
        Countries which contribute less than 1% of the revisions to
        the target page are accumulated together and labeled 'Other'.

        contribs      Map of contributions from each country.
        nRevs          Total number of revisions (user revs + anon revs)
        pageTitle      Title of the target page
        '''
        ccs = np.array(contribs.keys())
        revs = np.array(contribs.values())

        idx = revs.argsort()
        idx = idx[::-1]
        ccs = ccs[idx]
        revs = revs[idx] / revs.sum() * 100

        pct = 1.00    # PCT = 0.01 (1%) contribution
        labels = ccs[revs>pct]
        fracs = revs[revs>pct]

        # Lump together contributions of countries with less than PCT
        labels = labels.tolist()
        labels.append('Other')
        otherW = 100 - fracs.sum()
        fracs = fracs.tolist()
        fracs.append(otherW)

        ipNote = 'IPs known for {:,} / {:,} revs'.format(sum(contribs.values()), nRevs)

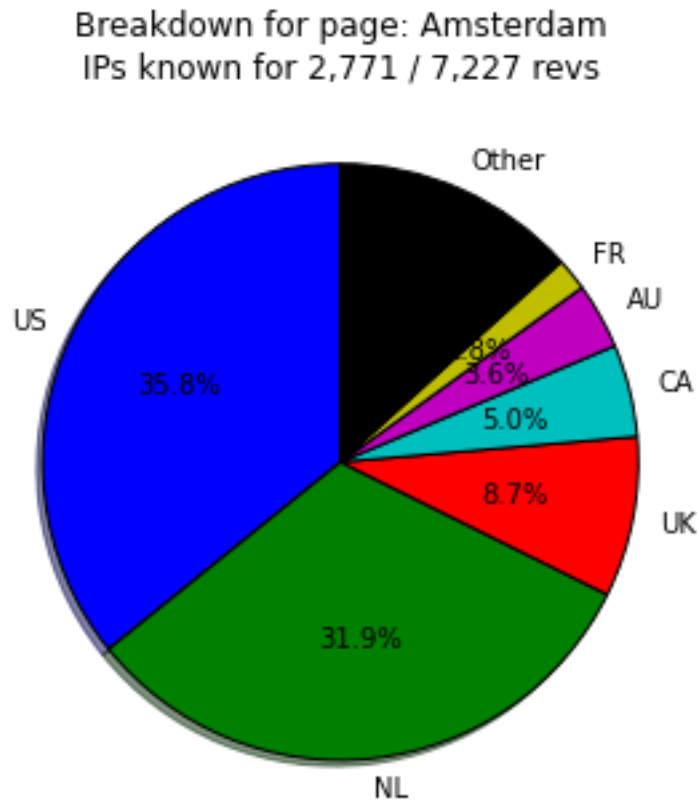
        figure(figsize=(5,5))
```

```
pie(fracs, labels=labels, autopct='%1.1f%%', shadow=True, startangle=90)
title('Breakdown for page: ' + pageTitle + '\n' + ipNote);
```

The following examples show the distribution of revisions from various countries to selected pages.

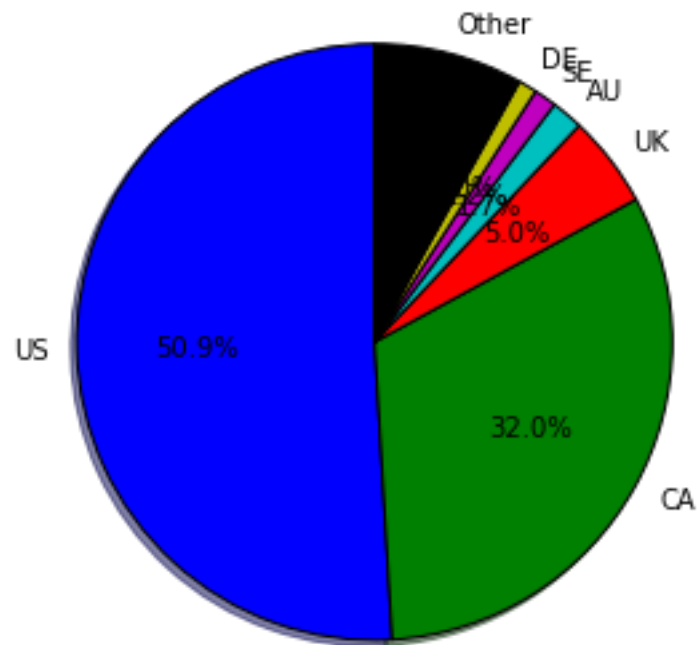
NOTE: These charts are based only on the known IP addresses (revisions from registered users are discarded).

```
In [11]: pageTitle = 'Amsterdam'
anonIPs, nRevs = getContributionsByIP(pageTitle)
contribs = countCountryContributions(anonIPs)
displayResults(contribs, nRevs, pageTitle)
```



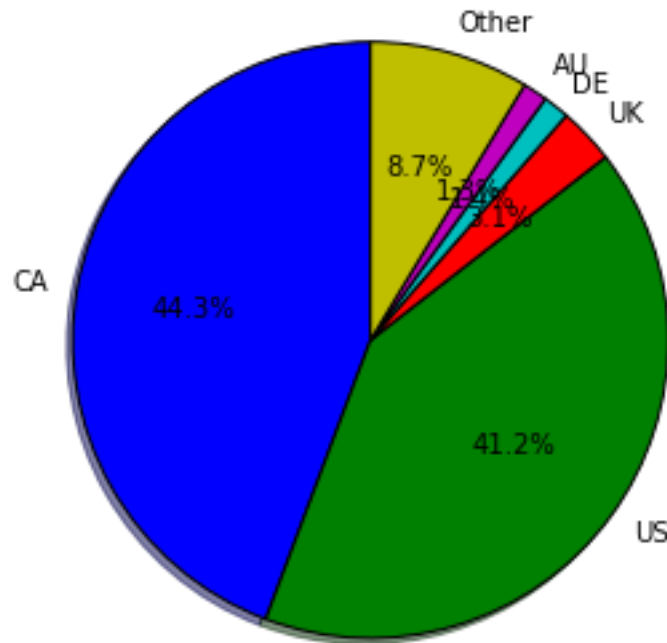
```
In [12]: pageTitle = 'Ice hockey'
anonIPs, nRevs = getContributionsByIP(pageTitle)
contribs = countCountryContributions(anonIPs)
displayResults(contribs, nRevs, pageTitle)
```

Breakdown for page: Ice hockey
IPs known for 3,981 / 8,698 revs



```
In [13]: pageTitle = 'Canada'  
anonIPs, nRevs = getContributionsByIP(pageTitle)  
contribs = countCountryContributions(anonIPs)  
displayResults(contribs, nRevs, pageTitle)
```

Breakdown for page: Canada
IPs known for 4,343 / 18,197 revs



1.2 Mwclient package

Mwclient python package provides an alternative to making queries from wikimedia API, basically wrapping the HTTP requests and JSON parsing so we don't have to worry about it.

```
In [14]: import mwclient
         wiki = mwclient.Site('en.wikipedia.org')
```

The *getContributionsByIP* functionality can be replicated using the Mwclient package. This functionality has been wrapped in the *btb.utils.wikiquery* python package.

```
In [15]: import btb.utils.wikiquery as wq
```

```
In [16]: ips, users, nRevs = wq.getContributionsForPage(wiki, 'Amsterdam')
```

```
print 'Revision history for page: ' + pageTitle
print '  Number of total revisions    : {:,}'.format(nRevs)
print '  Number of anonymous revisions: {:,}'.format(len(ips))
print '  Contribution IP\'s (first 10) : '
print '      ',ips[:10]
```

Revision history for page: Canada

Number of total revisions : 7,227

Number of anonymous revisions: 2,771

Contribution IP's (first 10) :

[u'108.254.160.23', u'2001:610:1908:C000:480D:5D05:FBB7:4AD5', u'173.77.169.32', u'217.120.225.80

1.3 Registered user contributions

So far we have been able to obtain country information from anonymous users, but we do not have information from registered users. Country information can be obtained from some registered users. The process for doing this is described in the [registered user contributions](#) notebook. Functionality for retrieving user information is contained in the `wikidat.utils.userresolver` module.

1.4 Other wikipedia data

The following is an overview of other useful functionality contained in the `btb.utils.wikiquery` package.

1.4.1 List bot users

Some wikipedia page revisions are performed by automated scripts known as *bots*. One may wish to ignore revisions made by bots and thus it is useful to have a list of all existing bots:

```
In [17]: bots = wq.getAllBots(wiki)

        print 'Currently {:,} bots are known to exist.'.format(len(bots))

{'augroup': 'bot'}
Currently 410 bots are known to exist.
```

1.5 Summary

The tools described here enable us to perform the following tasks:

- List IP's and usernames which contributed to a page
- Resolve the country of a given IP
- Resolve the country of a given username (if known)
- List all bot users

These tools will enable us to obtain the total contribution from each country to any given page, as described in [this](#) notebook.