Getting Data from the Web with R

Part 8: Getting Data via Web APIs

Gaston **S**anchez

April-May 2014

Content licensed under CC BY-NC-SA 4.0

Readme

License:

Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
http://creativecommons.org/licenses/by-nc-sa/4.0/

You are free to:

Share — copy and redistribute the material

Adapt — rebuild and transform the material

Under the following conditions:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made.

NonCommercial — You may not use this work for commercial purposes.

Share Alike — If you remix, transform, or build upon this work, you must distribute your contributions under the same license to this one.

Lectures Menu

Slide Decks

1. Introduction
2. Reading files from the Web
3. Basics of XML and HTML
4. Parsing XML / HTML content
5. Handling JSON data
6. HTTP basics and the RCurl package
7. Getting data via Web Forms
8. **Getting data via Web APIs**

CC BY-SA-NC 4.0

Web APIs

 CC BY-SA-NC 4.0

Goal

Data via Web APIs

The goal of the present slides is to give you an overview of **how data can be accessed via APIs** with R

 CC BY-SA-NC 4.0

Synopsis

In a nutshell

We'll cover the following topics:

- API basics
- API consederations
- Case Study PubMed

 CC BY-SA-NC 4.0

Some References

- XML and Web Technlogies for Data Sciences with R
  by Deb Nolan and Duncan Temple Lang
- RESTful Web Services
  by Leonard Richardson and Sam Ruby
- The ROAuth Package
  `http://cran.r-project.org/web/packages/ROAuth/index.html`
- CRAN Task View: Web Technologies and Services
  `http://cran.r-project.org/web/views/WebTechnologies.html`

CC BY-SA-NC 4.0

API Basics

 CC BY-SA-NC 4.0

API

"In computer programming, an application programming interface (API) specifies how some software components should interact with each other."

"When used in the context of web development, an API is typically defined as a set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages"

`http://en.wikipedia.org/wiki/Application_programming_interface`

 CC BY-SA-NC 4.0

About APIs

What's an API?

API stands for **Application Programming Interface**. Broadly speaking, API refers to a set of programming instructions that allow different software to interact with another.

Web API?

A Web API uses HTTP requests to access information from Web-based software application.

 CC BY-SA-NC 4.0

About APIs

What's an API?

By definition, an API is an interface, that is, something that **defines the way in which two software communicate**.

How Web APIs communicate?

With Web APIs, the communication between applications is handled through a collection of standards and protocols (eg HTTP)

 CC BY-SA-NC 4.0

Importance

Why should we care?

A lot of companies allow users to freely access data from their websites by means of APIs. So it is important to have some exposure on how to work and get data with APIs.

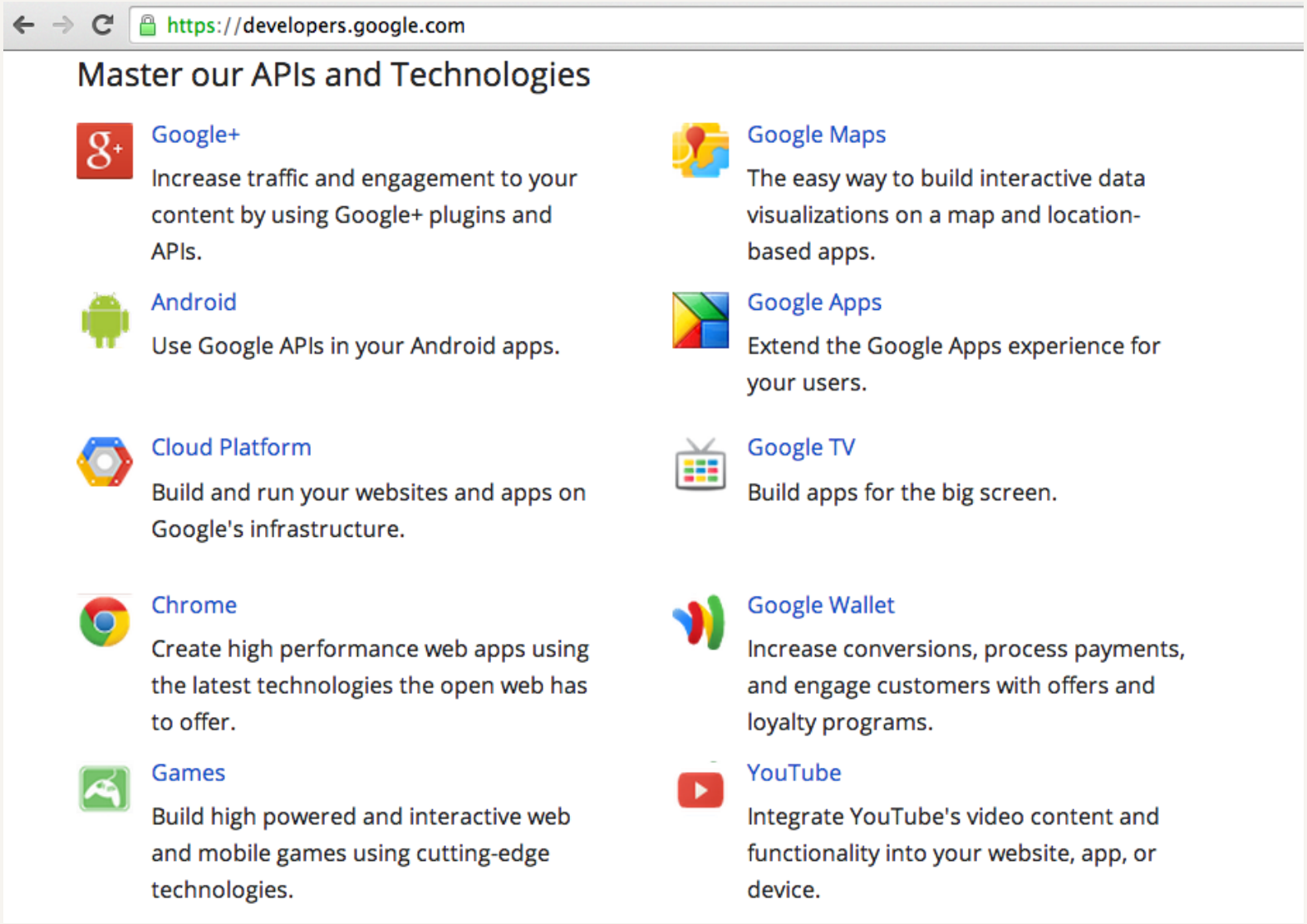
Everybody who's anybody has an API

Web APIs are becoming increasingly popular for any company, almost mandatory if they want to stay "connected" with the rest of the world, and have a presence with their users/customers. A good starting point for finding more websites with APIs is **Programmable Web**:

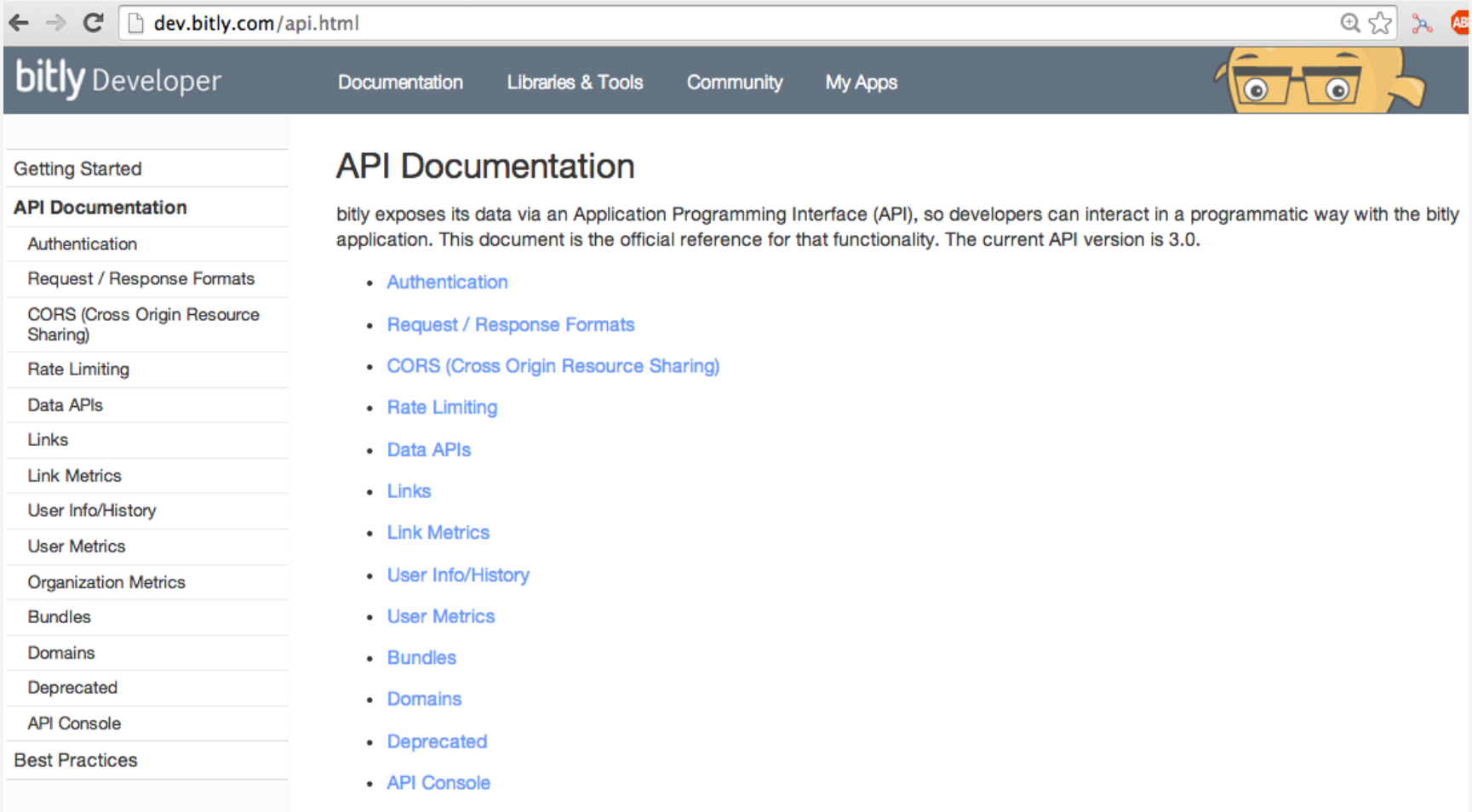
`http://www.programmableweb.com`

 CC BY-SA-NC 4.0

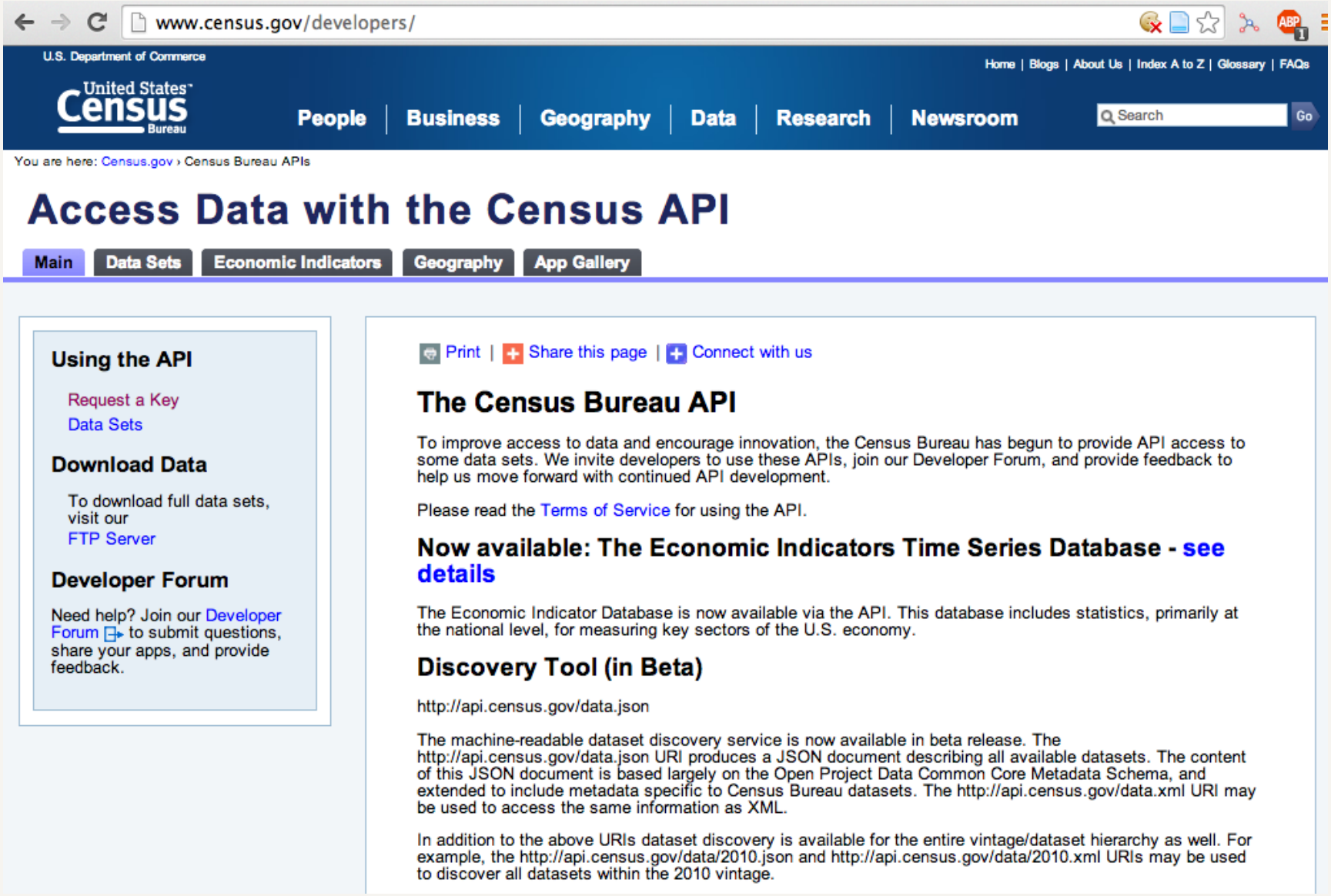
Web API Examples: Google APIs



Web API Examples: Bitly



Web API Examples: US Census



Web API Considerations

 CC BY-SA-NC 4.0

Considerations

We're just here for the data

We're interested in Web APIs because they allow us to download content and data. And as with any other tool, there's the good, the bad, and the ugly about Web APIs.

 CC BY-SA-NC 4.0

Considerations

The Good

- Obviously, we get access to data
- No need to buy or pay for data
- We can get access in a programmatically way
- There are some R packages for working with specific Web APIs

  `http://cran.r-project.org/web/views/WebTechnologies.html`
- Under ideal circumnstances, data is in nice formats and decent structure

CC BY-SA-NC 4.0

Considerations

The Bad

- You first have to learn how to use a given Web API
- Learning about an API and reading the documentation may take you some time
- The documentation may be very limited, outdated, and poorly explained
- Sometimes the data is not what you were expecting

 CC BY-SA-NC 4.0

Considerations

The Ugly

- APIs change and evolve, so if you're planning to use one over a long period of time, you have to check it constantly
- You may need to register, get credentials, use authentication
- Usually there are rate and quota limits on how much data you can download and how frequently
- Some APIs can be very complex to use (not user friendly)

 CC BY-SA-NC 4.0

Working with Web APIs

Checklist for working with Web APIs

- Read the API's documentation (check the examples!)
- Read terms of service (what you're allowed or not to do)
- Typically you may need an API key
- You may need to register and open a developer account
- You may need to authenticate (just another step in the pipeline)
- See what kind of data you have access to
- See what kind of formats are available

 CC BY-SA-NC 4.0

Working with Web APIs and R

This is our main concern: figure out how to interact with Web APIs and do things in R

- Understand how the API works
  (eg REST, SOAP)
- Understand what kind of requests you need to use
  (GET, POST, etc)
- Experiment first with simple examples
- Usually you begin with some kind of query or search
- Check what R packages you must load
  (eg XML, RCurl, SSOAP, ROAuth, httpRequest, RJSONIO)
- Write functions and scripts (programmatically)

 CC BY-SA-NC 4.0

Case Study
E-utilities with PubMed

 CC BY-SA-NC 4.0

Case Study Outline

Outline

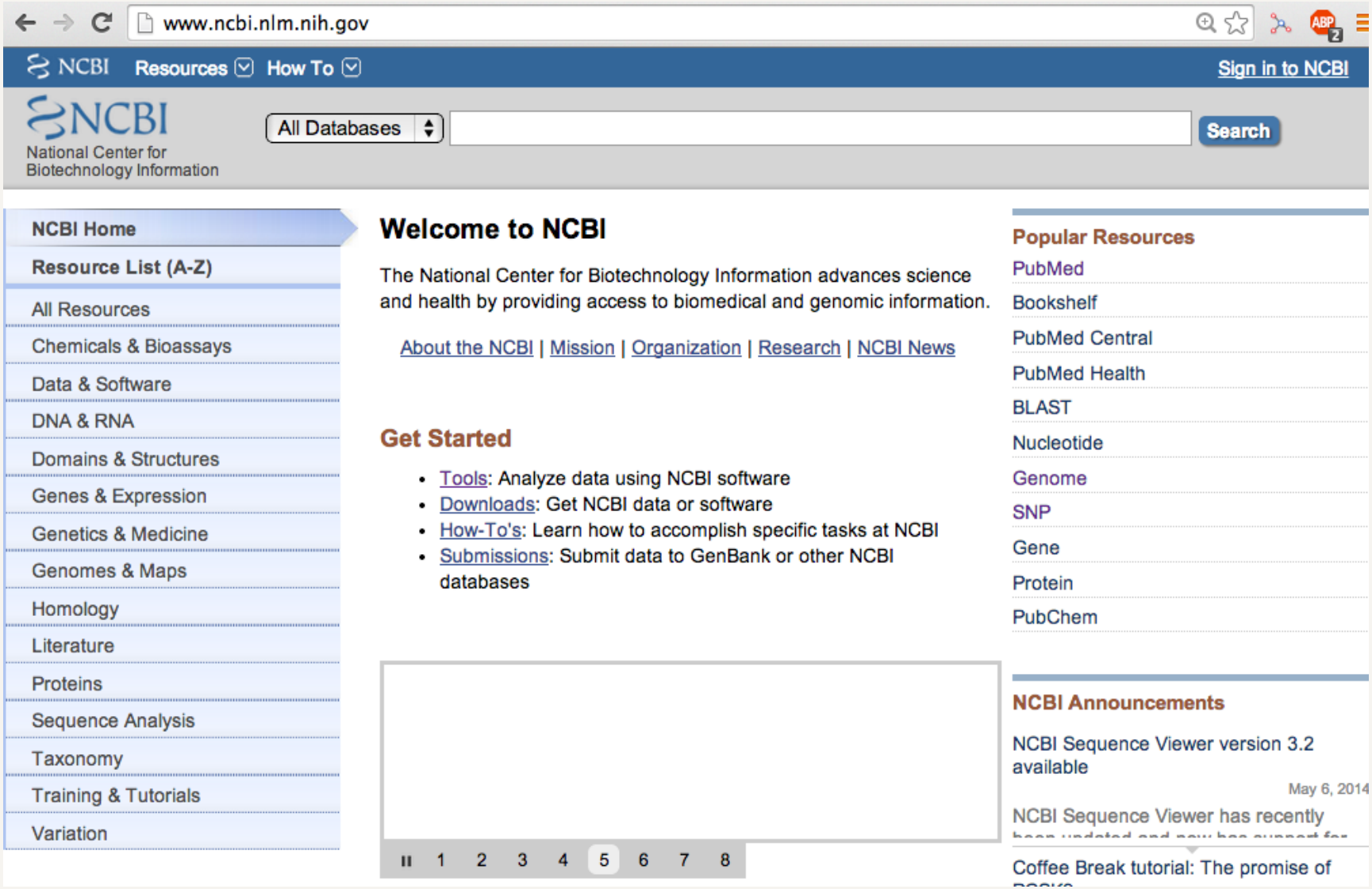
We'll work with a **small but real life example**, which means you might get lost with all the descriptive information.

Hopefully this brief outline will help you to navigate through this case study:

- **NCBI** Website (central site containing everything else)
- Database **PubMed** (this is where we want to get data from)
- Web API **E-Utilities** (this is the API we'll be playing with)
- **ESearch** and **ESummary** (in particular, we'll use these 2 *functions*)

CC BY-SA-NC 4.0

National Center for Biotechnology Information



About NCBI

NCBI

The **National Center for Biotechnology Information** (NCBI) provides access to biomedical information

`http://www.ncbi.nlm.nih.gov`

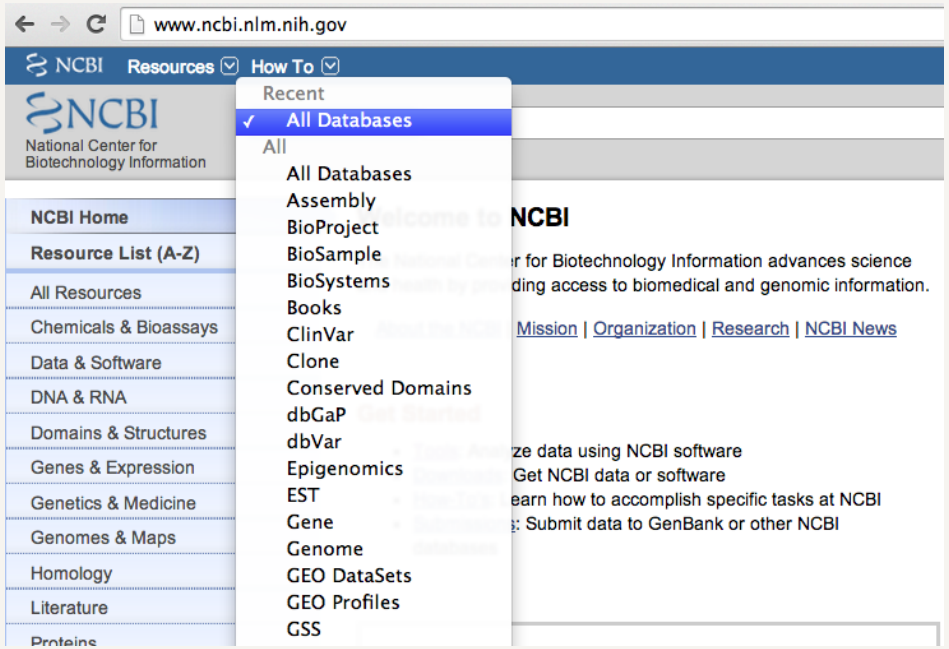
Mind you

- The NCBI website is kind of ugly and confusing.
  (I told you this is a real life example!)
- Looking at the website, it's not evident what you are supposed to do first or where to begin with.
- For our example, the important parts of the website are the **Databases** and the **Tools**.

About NCBI

NCBI Databases

The real juice of the NCBI website is the data in it. There are **38 databases** covering a variety of biomedical data. Among them, there is the database *PubMed*.

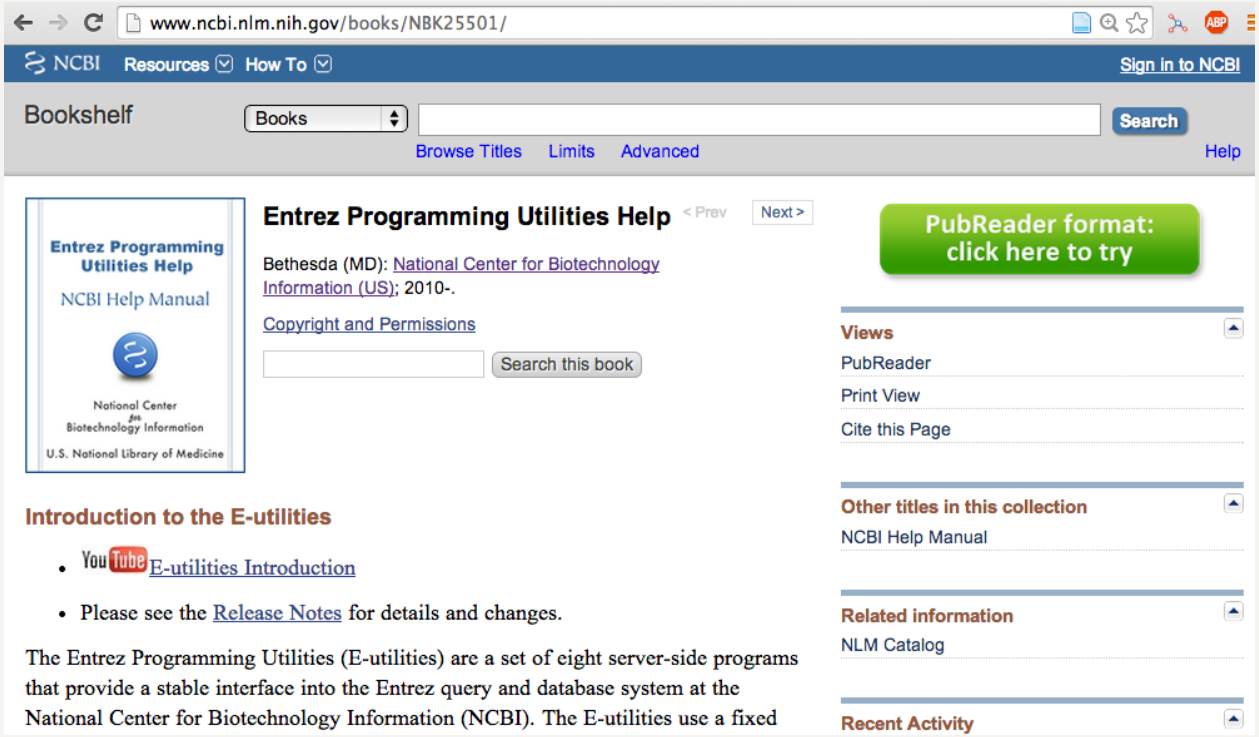


API: E-Utilities

E-Utilities

The NCBI makes accessible the information in their databases with its API **Entrez Programming Utilities** also known as **E-Utilities**

http://www.ncbi.nlm.nih.gov/books/NBK25501/

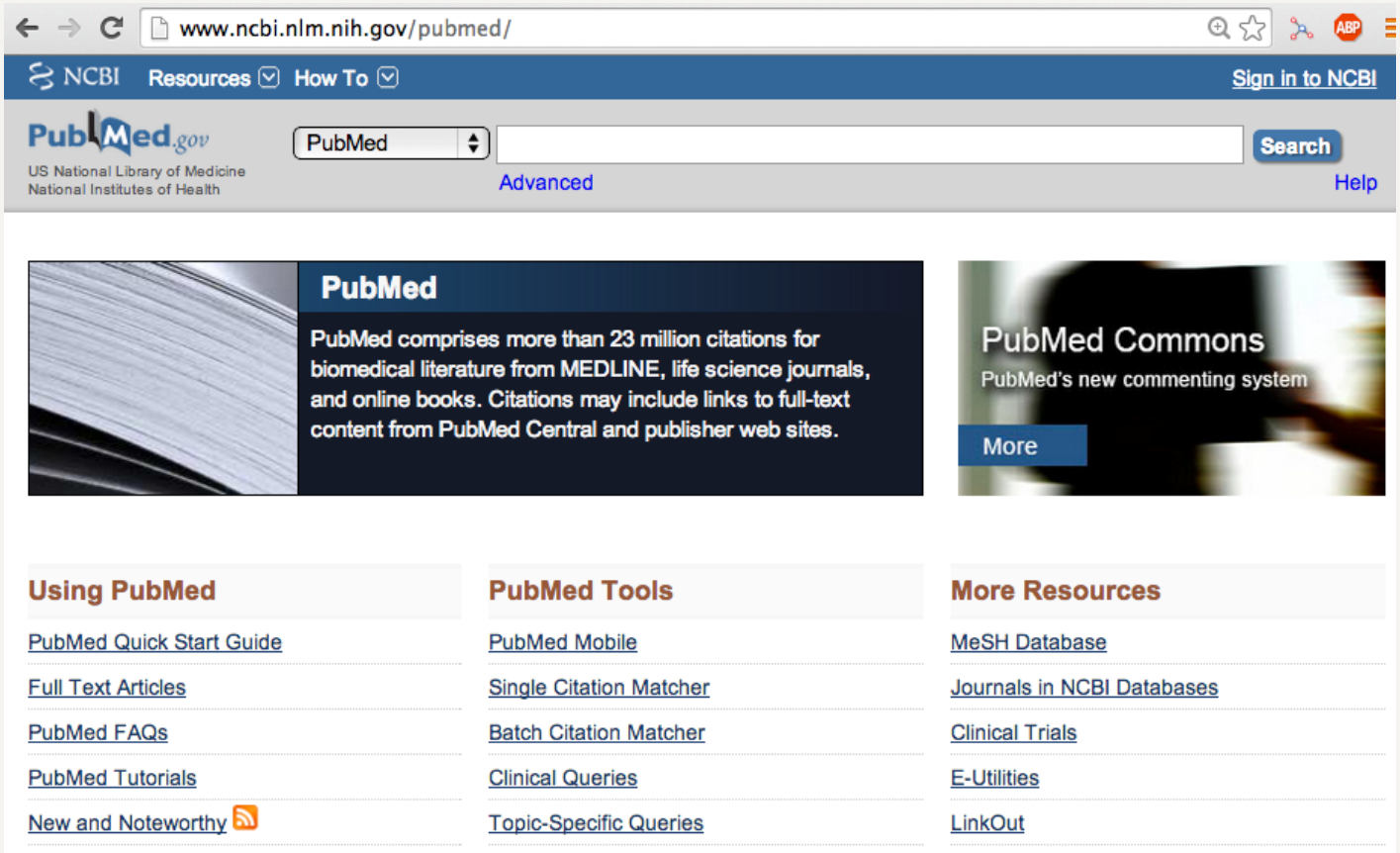


About PubMed

PubMed Database

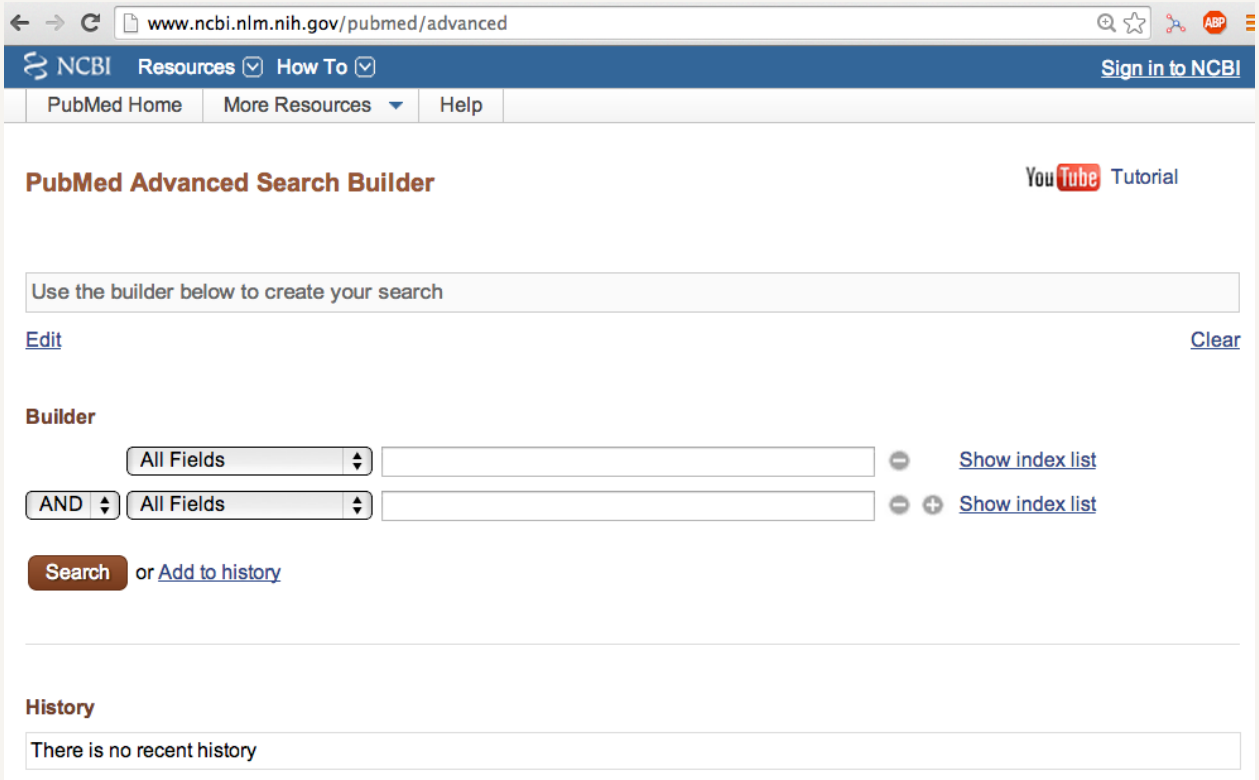
PubMed comprises more than 23 million citations for biomedical literature from MEDLINE, life science journals, and online books.

http://www.ncbi.nlm.nih.gov/pubmed/



Searching PubMed

Although we can use the Advanced Search forms to query PubMed (`http://www.ncbi.nlm.nih.gov/pubmed/advanced`), we'll see how to use the E-utilities API to get some data.



Mission

Ultimate Goal

Your mission is to download the PubMed IDs for articles published in 2012, and containing the term *"human genome"* in their titles

Good Luck!

 CC BY-SA-NC 4.0

Basics of E-utilities

(NCBI Web API)

 CC BY-SA-NC 4.0

E-Utilities

E-Utilities provides 8 different applications or utilities:

Application	Description
`EInfo`	database statistics
`ESearch`	text searches
`EPost`	Unique Identification (UID) uploads
`ESummary`	document summary downloads
`EFetch`	data record downloads
`ELink`	Entrez links
`EGQuery`	global query
`ESpell`	spelling suggestions
`ECitMatch`	batch citation searching in PubMed

We'll focus on *ESearch* and *ESummary*

 CC BY-SA-NC 4.0

E-Utilities Documentation

Documentation

Like many other Web APIs, there's an extensive documentation about E-Utilities ... and it's not beginner friendly.

But this is a good example of the things you will find when working with APIs. Usually you will struggle reading and understanding the provided documentation.

 CC BY-SA-NC 4.0

E-Utilities Applications

How does E-utilities work?

In essence, we use E-utilities by specifying URL's and send requests

1. All E-utilities calls share the same URL:
   `http://eutils.ncbi.nlm.nih.gov/entrez/eutils/`
2. Each E-utility has its own structure and parameters:
   - `esearch.fcgi?db=<database>&term=<query>`
   - `esummary.fcgi?db=<database>&id=<uid_list>`
3. All E-utilities are appended to the base url:

   `http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=<database>&term=<query>`

CC BY-SA-NC 4.0

ESearch and ESummary

ESearch & ESummary 2-step process

The way you'll work with ESearch and ESummary is using a two-step procedure:

1. you **search for a particular query** in order to get a list of UIDs (unique identification). This is done with **ESearch**
2. you use the list of UIDs to get a summary of the resources you are looking for. This is done with **ESummary**

 CC BY-SA-NC 4.0

ESearch

About ESearch

ESearch allows us to specify a **search query on a given database**. An ESearch call has the following structue:

`esearch.fcgi?db=<database>&term=<query>`

Parameters

- `<database>` represents the name of the database
- `<query>` represents the terms forming the query

 CC BY-SA-NC 4.0

Defining an ESearch Call

Example

We want to searh *PubMed* and get a list of UIDs for those articles containing the term *human genome* in the title, and that were published in *2012*.

Parameters

database: `pubmed`

query: `human+genome[title]+AND+2012[pdat]`

ESearch Call

```
esearch.fcgi?db=pubmed&term=human+genome[title]+AND+2012[pdat]
```

 CC BY-SA-NC 4.0

ESearch with R

```
# load XML
library(XML)

# E-utilities base url
base_url = "http://eutils.ncbi.nlm.nih.gov/entrez/eutils/"

# define database variable: PubMed
db = "pubmed"

# define query variable
query = "human+genome[title]+AND+2012[pdat]"

# assemble ESearch call
esearch = sprintf("esearch.fcgi?db=%s&term=%s", db, query)

# create URL to make search request
search_url = paste(base_url, esearch, sep = '')

# ESearch request to get the list of UIDs
# (the list will be contained in an XML document)
search_doc = xmlParse(search_url)
```


CC BY-SA-NC 4.0

ESearch XML file output

```
<?xml version="1.0" ?>
<!DOCTYPE eSearchResult PUBLIC ...>
<eSearchResult>
  <Count>103</Count>
  <RetMax>20</RetMax>
  <RetStart>0</RetStart>
  <QueryKey>1</QueryKey>
  <WebEnv>NCID_1_282865897_130.14.18.34_9001_1399488846_694838780</WebEnv>
  <IdList>
    <Id>23959643</Id>
    <Id>23730202</Id>
    ...
  </IdList>
   ...
</eSearchResult>
```

There's a `Count` of 103 articles. Each article has its corresponding `Id`. By default (`RetMax`) just 20 Ids are retrieved

 CC BY-SA-NC 4.0

ESearch with R

In order to get the 103 articles, we need to change the default maximum number of retrieved results:

```
# let's add the retmax parameter
retmax = 200

# redefine assemble ESearch
new_esearch = sprintf("esearch.fcgi?db=%s&term=%s&retmax=%s", db, query, retmax)

# update URL to make request
new_search_url = paste(base_url, new_esearch, sep = '')

# ESearch request to get the list of UIDs
# (the list will be contained in an XML document)
new_search_doc = xmlParse(new_search_url)
```

 CC BY-SA-NC 4.0

ESearch with R

We need to get the list of <Id>'s that are in the XML node <IdList>. One way to get those values is by using `xpathSApply()`

```
# get UIDs in a vector
ids = xpathSApply(new_search_doc, path = "//IdList/Id", fun = 'xmlValue')

# we should have 103 IDs
length(ids)


## [1] 103


# take a peek
head(ids)


## [1] "23959643" "23730202" "23520917" "23281708" "23281599" "23266811"


tail(ids)


## [1] "22057813" "22057783" "22050290" "22031941" "21945885" "21559844"
```


CC BY-SA-NC 4.0

ESummary

About ESummary

ESummary allows us to get *document summaries* for a list of input UIDs. An ESummary call has the following structure:

`esummary.fcgi?db=<database>&id=<uid_list>`

Parameters

- `<database>` represents the name of the database
- `<uid_list>` represents the list of UIDs

 CC BY-SA-NC 4.0

ESummary

Example

With ESearch we retrieve the list of UIDs for the articles we're looking for. Now we need to pass the list to ESummary

Parameters

database: `pubmed`
uid_list: `23959643,23730202,23520917,...`

ESummary Call

`esummary.fcgi?db=pubmed&id=23959643,23730202,23520917,...`

CC BY-SA-NC 4.0

ESummary with R

Now we need to use ESummary to get summaries of each article:

```
# concatenate all IDs, separated by commas
# (we'll pass this list to ESummary)
id_list = paste(ids, collapse = ',')

# assemble ESummary call
esummary = sprintf("esummary.fcgi?db=%s&id=%s", db, id_list)

# create URL to make request
sum_url = paste(base_url, esummary, sep = '')

# ESummary request to get document summaries
# (the summaries will be contained in an XML document)
summary_doc = xmlParse(sum_url)
```

 CC BY-SA-NC 4.0

ESummary output

You should get something like this from ESummary:

```
## <?xml version="1.0" encoding="UTF-8"?>
## <!DOCTYPE eSummaryResult PUBLIC "-//NLM//DTD esummary v1 20060131//EN" "http://eutils.ncbi.nlm.nih.gov/e
## <eSummaryResult>
## <DocSum>
##  <Id>23959643</Id>
##  <Item Name="PubDate" Type="Date">2012 Dec</Item>
##  <Item Name="EPubDate" Type="Date"></Item>
##  <Item Name="Source" Type="String">Hum Biol</Item>
##  <Item Name="AuthorList" Type="List">
##  <Item Name="Author" Type="String">Casto AM</Item>
##  <Item Name="Author" Type="String">Henn BM</Item>
##  <Item Name="Author" Type="String">Kidd JM</Item>
##  <Item Name="Author" Type="String">Bustamante CD</Item>
##  <Item Name="Author" Type="String">Feldman MW</Item>
##  </Item>
##  <Item Name="LastAuthor" Type="String">Feldman MW</Item>
##  <Item Name="Title" Type="String">A tale of two haplotypes: the EDA2R/AR Intergenic region is the most c
##  <Item Name="Volume" Type="String">84</Item>
##  <Item Name="Issue" Type="String">6</Item>
##  <Item Name="Pages" Type="String">641-94</Item>
##  <Item Name="LangList" Type="List">
##  <Item Name="Lang" Type="String">English</Item>
##  </Item>
##  <Item Name="NlmUniqueID" Type="String">0116717</Item>
##  <Item Name="ISSN" Type="String">0018-7143</Item>
```

Journal Names

The summary document of each <Id> article is composed of several <Item>'s such as "PubDate", "AuthorList", "Title", "Lang", and "FullJournalName", among others.

The **journal name** of each article is in an element:
`<Item Name="FullJournalName" Type="String">`

For instance, the first article of the list (<Id>23959643</Id>) was published in *Human Biology*:
<Item Name="FullJournalName" Type="String">Human biology</Item>

 CC BY-SA-NC 4.0

ESummary with R

If we want to get the journal names in which each article was published, we can use `xpathSApply()` like so:

```
# journal names
journals = xpathSApply(summary_doc, "//Item[@Name='FullJournalName']", xmlValue)

# first 5 journals
head(journals, n = 5)


## [1] "Human biology"
## [2] "Current genomics"
## [3] "Revista de derecho y genoma humano = Law and the human genome review / Catedra de Derecho y Genoma
## [4] "BMC genomics"
## [5] "BMC genomics"


# last 5 journals
tail(journals, n = 5)


## [1] "Human genetics"              "Tissue antigens"
## [3] "Journal of virology"         "Human molecular genetics"
## [5] "Journal of medical systems"
```


CC BY-SA-NC 4.0