

Wikipedia data analysis.
Introduction and practical examples

Felipe Ortega

June 29, 2012

Abstract

This document offers a gentle and accessible introduction to conduct quantitative analysis with Wikipedia data. The large size of many Wikipedia communities, the fact that (for many languages) the already account for more than a decade of activity and the precise level of detail of records accounting for this activity represent an unparalleled opportunity for researchers to conduct interesting studies for a wide variety of scientific disciplines.

The focus of this introductory guide is on data retrieval and data preparation. Many research works have explained in detail numerous examples of Wikipedia data analysis, including numerical and graphical results. However, in many cases very little attention is paid to explain the data sources that were used in these studies, how these data were prepared before the analysis and additional information that is also available to extend the analysis in future studies. This is rather unfortunate since, in general, data retrieval and data preparation usually consumes no less than 75% of the whole time devoted to quantitative analysis, specially in very large datasets.

As a result, the main goal of this document is to fill in this gap, providing detailed descriptions of available data sources in Wikipedia, and practical methods and tools to prepare these data for the analysis. This is the focus of the first part, dealing with data retrieval and preparation. It also presents an overview of useful existing tools and frameworks to facilitate this process. The second part includes a description of a general methodology to undertake quantitative analysis with Wikipedia data, and open source tools that can serve as building blocks for researchers to implement their own analysis process. Finally, the last part presents some practical examples of quantitative analyses with Wikipedia data. In each of these examples, the main objectives are first introduced, followed by the data and tools required to conduct the analysis. After this, the implementation and results of the analysis are described, including code in SQL, Python and R to carry out all necessary actions. All examples conclude with pointers to further references and previous works for readers interested in learning more details.

I hope that this guide can help some researchers to enter the fascinating world of Wikipedia research and data analysis, getting to know available data sources and practical methodologies so that they can apply their valuable expertise and knowledge to improve our understanding about how Wikipedia works.

Contents

I	Preparing for Wikipedia data analysis	3
1	Understanding Wikipedia data sources	4
1.1	Public data sources	4
1.2	Different languages, different communities	5
1.3	Activity vs. traffic	5
1.4	Web content	6
1.5	MediaWiki API	6
1.6	The toolserver	7
1.7	Wikipedia dump files	7
1.7.1	Types of dump files	9
1.7.2	Complete activity records: <i>stub-meta</i> and <i>pages-meta</i>	10
1.7.3	User groups	10
1.7.4	Logging dumps: administrative and maintenance tasks	11
2	Data retrieval, preparation and storage	16
2.1	RSS to notify updates	16
2.2	The particular case of the English Wikipedia	17
2.3	Computing extra metadata	17
2.4	Practical tips and assessment	18
2.4.1	DRY and ARW	18
2.4.2	Data storage: hardware	18
2.4.3	Operating system and file system support	21
2.4.4	Data storage: database engines	22
3	Available tools for Wikipedia data extraction and analysis	25
3.1	Here be dragons	25
3.2	Wikistats and report cards	26
3.3	StatMediaWiki and Wikievidens	26
3.4	Interacting with the MediaWiki API	27
3.4.1	Pywikipediabot	27
3.4.2	Python-wikitoools	28
3.4.3	Mwclient	28
3.5	WikiTrust (for data analysis)	29
3.6	Pymwdat	30
3.7	Wikimedia-utilities	31
3.8	WikiDAT: Wikipedia Data Analysis Toolkit	31

II	Conducting Wikipedia data analysis	36
4	Methodology for Wikipedia data analysis	37
4.1	The big picture	37
4.2	Retrieve and store your data locally	38
4.3	Routinary tasks for data cleaning and preparation	39
4.4	Know your dataset	40
5	Open source tools for data analysis	42
5.1	Python	42
5.1.1	Installing Python in GNU/Linux	42
5.1.2	Installing Python in Windows or Mac OS	43
5.1.3	The <i>mysql-python</i> API	43
5.2	NumPy, SciPy and matplotlib	43
5.3	Python Scikit-learn	43
5.4	Database engine: MySQL	44
5.5	R programming language and environment	44
5.5.1	Installing R	44
5.5.2	Installing additional R libraries	45
5.5.3	Graphical user interfaces for R	46
5.5.4	R documentation and further references	46
6	Example cases in Wikipedia data analysis	48
6.1	Activity metrics	48
6.1.1	Questions and goals	48
6.1.2	Required data and tools	48
6.1.3	Conducting the analysis	49
6.1.4	Further reading	49
6.2	The study of inequalities	49
6.2.1	Questions and goals	49
6.2.2	Required data and tools	50
6.2.3	Conducting the analysis	50
6.2.4	Further reading	50
6.3	The study of <i>logging</i> actions	50
6.3.1	Questions and goals	50
6.3.2	Required data and tools	50
6.3.3	Conducting the analysis	50

Part I

Preparing for Wikipedia data analysis

Chapter 1

Understanding Wikipedia data sources

The first step in data analysis is always to find an interesting question or a set of questions (sometimes hypotheses) that we want to answer or validate. Then, we search out for data to answer these questions or assess the validity of our hypotheses.

Once we have found promising data sources, we need to collect these data, load or store them somewhere, and start the exploratory phase of data analysis. In this stage, we try to compute numerical and graphical descriptors of our data set to validate its adequacy for the purposes of our study. In this stage, we must also pay attention to detect any special features in our data set (missing values, extreme values, incorrect labeling, etc.) that may threaten the validity of our analysis.

It is very curious to check how very little attention is frequently devoted to this important stage of data analysis. Space limitations in many types of research publications (journals, tech reports, conference papers, etc.) usually force all authors to concentrate on the analysis, results and discussion part, which are the sections delivering the actual scientific contribution. However, this also makes quite difficult for other researchers to follow these studies, replicate their results and extend the analysis beyond its initial scope. This is particularly true for analysis using very large data sets, such as in many studies involving Wikipedia data.

Over the past 5 years, I have reviewed many scientific works and reports dealing with Wikipedia data analysis. To my surprise, I still can find many problems linked to incorrect assumptions about the nature of the process generating the data, their meaning, problems with data preparation (unfiltered items introducing noise or that are irrelevant for the purposes of the study), and many other issues that jeopardize the validity of quite interesting research works.

My aim in this guide is to (hopefully) help you to avoid committing these common pitfalls in your own studies. To achieve this, the first step is to learn all necessary details about Wikipedia data and available information sources that you can use to retrieve it.

1.1 Public data sources

The first important characteristic of Wikipedia data sources is that many useful data records are publicly available for download and use. This include information about Wikipedia content, discussion pages, user pages, editing activity (who, what and when) administrative tasks (reviewing content, blocking users, deleting or moving pages), and many, many other details.

The main exception to this general rule is private data about Wikipedia users who registered in any of the more than 280 different languages available in Wikipedia (more on this in the next section). This is for a very good reason: it is not my business or your business to learn

about the real name or e-mail address of Wikipedia users. Privacy must be preserved and Wikimedia Foundation does a splendid job in this regard. In any case, people use a nickname that is associated to their user account, and that is displayed as their public identifier in the community. For the purpose of tracking individual contributions, this should suffice for most of your studies.

At the same time, this also means that Wikipedia offers detailed records about all the activity performed by users, making it a public forum compliant with the design principles of *social translucence* [[TODO:ref]] in digital systems. Of course, that means that we can virtually track the complete activity history of every single user in Wikipedia. However, from a research ethics point of view, despite many users are fully aware that their digital footsteps can be traced in detail, some of them may not feel very comfortable if they are pinpointed in a study. So as a researchers, we also have the responsibility of reporting this information with care, and avoid unnecessary precise details in our results for the sake of respecting some of the *public privacy* of Wikipedia users (in particular, without obtaining their explicit consent).

In summary, Wikipedia offers a vast and quite rich data garden for researchers, which is public and made available through some convenient channels that we are going to discover in the next sections. So, let's start our walk through the Wikipedia data store.

1.2 Different languages, different communities

Another important trait of Wikipedia data is that a different MediaWiki site is maintained for every different language in Wikipedia. Therefore, in case that a user wants to contribute to more than one language with a user account, she must register for a new account in each language. Now, a unified login feature has been deployed to facilitate users with presence in multiple language to perform simultaneous login in all of them in a single step ¹. However, in many cases users find that their nickname in one language has already been taken by other person in a different language. For older accounts, this is a very common scenario.

As a result of this, in general we must stick with the assumption that nicknames in Wikipedia identify users only within a certain language community, and that they do not identify users consistently across different languages, unless we can prove otherwise reliably.

Besides, having different communities also means that they may follow distinct policies or conventions to organize their online work. Admittedly, in all cases this does not include the very basic organizational principles underpinning the whole Wikipedia project, such as *the five pillars* ². Nevertheless, this does include some variations that may alter substantive characteristics of the activity that generates Wikipedia data and metadata in a given language. For instance, whereas in many languages users promoted to admin status retain it for life (unless it is revoked for good reasons), in the Norwegian Wikipedia they must renew this status periodically. Another example is the habit extended in the Polish Wikipedia of deleting talk pages that remain idle for some time, thus inducing us to infer (somewhat incorrectly) that their interesting in debates on articles contents is abnormally low.

1.3 Activity vs. traffic

Yet another important remark is that Wikipedia not only offers data about *actions that change the status of the system* (edits, page moving or deletion, blocking users, etc.) but also about *requests that do not alter the system status or content*. A clear example of the latter is page views

¹http://en.wikipedia.org/wiki/Wikipedia:Unified_login

²http://en.wikipedia.org/wiki/Wikipedia:Five_pillars

(requests to Wikipedia servers to display a wiki page on our browser). Further examples are searches performed on the site or clicking on the edit or preview button while changing the content of a wiki page (but without clicking on the save button, which is the action actually recorded on the system database).

We can call this data about requests that do not modify the status of the system *traffic data*, while *activity data* will refer to actions actually modifying the status of the system. We are not going to cover traffic data in this guide. The main reason is that only some of this data, namely for page views, is publicly available ³. However, in case that you are interested in obtaining a sampling for all possible actions in Wikipedia traffic data you must contact the *Wikipedia Research Committee* ⁴ directly to ask for a data stream containing that sample. This data stream will be, in all cases, conveniently anonymized to elid all private information about users performing requests to the system.

In case that you are interested in learning more about the type of analyses that we can conduct with this data, I recommend you to grab a good cup of your favourite stimulating beverage and take a look at the doctoral thesis written by my colleague Antonio J. Reinoso [[TODO: citation]] about Wikipedia traffic ⁵.

1.4 Web content

OK, so finally we start to describe some available sources to retrieve Wikipedia data. The first, and probably most obvious one, is to obtain this data from the information displayed on our browser when we request any Wikipedia page. Some people could think that retrieving information about Wikipedia articles in this way, sometimes known as *web scraping* ⁶ is a good procedure to accomplish this.

I certainly discourage you to follow this path, unless you have very strong and good reasons supporting it. The main argument against this approach is that, if you try to visit to many pages at a very fast pace to retrieve their content, high chances are that you get banned by Wikipedia's system centries on the premise of creating excessive traffic load. Moreover, as we are goint to learn in the next sections, there are many alternative ways providing you much more information without the downside of overloading the Wikipedia side with a zillion of requests.

1.5 MediaWiki API

A very useful data retrieval channel for real-time requests lovers is to use the MediaWiki API ⁷, available for most (if not all) of the Wikipedia languages. Like many other data retrieval APIs, it offers you a structured syntax to query the Wikipedia system for the data we are looking for. Requests take the form of HTTP GET requests accepting numerous possible parameters to refine the terms of our petition to the system.

A big advantage of using the MediaWiki API is that it also offers a `format` parameter that let consumers control the format in which the answer is represented. Available output formats

³<http://stats.grok.se/>

⁴<http://meta.wikimedia.org/wiki/Research:Committee>

⁵By no means I meant with this that reading Antonio's thesis is a boring activity. He did quite a remarkable work and it is written in clear style. All the same, reading any PhD. dissertation is always a daunting task, as you are often exposed to far too many information and fine grain details that you usually need. May the Force be with you.

⁶http://en.wikipedia.org/wiki/Web_scraping

⁷http://www.mediawiki.org/wiki/API:Main_page

include the most important data representation standards, like JSON, WDDX, XML, YAML or PHP native serializaton format. This is also a good way to query the MediaWiki database for obtaining answers to precise questions or subsets of data matching certain conditions ⁸. I am not convering yet the many data fields available in the database server, since they will be described in the section about the dump files, later on.

On the other side, we must keep in mind that we shouldn't push very hard this data retrieval channel, specially since some of the queries must be expensive for Wikipedia servers to attend in terms of system resources. Admins reserver the right to ban any user that may abuse this service ⁹. Thus, this is a very flexible a convenient way to peform concrete questions to the system when we know in advance that the answers should not return very large data sets. It can also be very useful for software applications that need to query the Wikipedia system in (almost) real-time. However, if you are interested in retrieving the complete activity history of the whole English Wikipedia, arm yourself with a high dosis of patience or better try one of the other available alternatives.

1.6 The toolserver

The Wikimedia toolserver ¹⁰ is a cluster of servers operated by Wikimedia Deutschland, one of the Chapters officially approved in the Wimipedia global movement. Its aim is to maintain replicas of databases from all Wikimedia's project, in order to test new software, provide added-value services and statistics based on these data, and undertake research activities. For example, some of the new statistics about Wikipedia articles are computed by software services running on the toolserver ¹¹.

In order to get access to this infrastructure, interested researchers must contact the toolserver responsables to obtain a user account and shell access. Before you ask, yes, the servers run on GNU/Linux, so you must have some minimal background about perfoming connections to remote systems on SSH and basic shell usage. The database engine to store this replicas is MySQL (the same engine used for all Wikimedia projects up to know).

The infrastructure is quite powerful, though the usual common sense and etiquette rules apply here, as well, regarding the use of common computational resources.

1.7 Wikipedia dump files

The best way to retrieve large portions or the whole set of data and metadata about any action performed in a Wikipedia language is using the database dump files. These dump files contain not only the wikitext but also metadata about all actions that have been recorded in the database of any Wikimedia wiki. The files are publicly available on the Wikimedia Downloads center ¹². Lately, they have also been replicated on some mirrors provided by several institutions around the world that offer storage space and bandwidth *pro bono* to help balancing the load.

These dump files usually have two possible formats for data representation:

- *XML files*: XML is the preferred format for large dump files, such as those storing information about changes in wiki pages, or administrative and maintentance actions per-

⁸<https://www.mediawiki.org/wiki/API:Query>

⁹<https://www.mediawiki.org/wiki/API:Etiquette>

¹⁰<http://toolserver.org>

¹¹<http://toolserver.org/~tparis/articleinfo/index.php>

¹²<http://dumps.wikimedia.org/>

formed in the wiki. XML presents a neutral format to export this information and recover it locally to another wiki system (for example, to build a replica of Wikipedia in a given language) or any other storage system.

- *SQL files*: SQL files are usually published for small or medium size dump files, like external or internal links in wiki pages, the list of redirects, or the list of special users and their associated privileges.

Since these dump files can consume a lot of storage capacity in the servers, it is a common practice to publish compressed version of these files, using different algorithms according to the compression rate required to reduce the size of the files within manageable limits. On one hand, very large dump files like the ones containing all changes recorded in large Wikipedia languages (German, French, etc.) are compressed in 7-zip or bzip2 format. On the other hand, smaller dump files are compress in gzip format, since it can be decompressed at a faster rate and we not need the same high compression ratio as in the larger dumps.

In the case of the English Wikipedia, the single dump file that was formerly produced containing all wiki text and metadata about changes recorded in the database has been conveniently split into multiple pieces or chunks. Each of these chunks or slices store information about all changes performed in a range of wiki pages. The limits of the range (in the form of the unique numerical identifiers assigned to every page) are indicated in the file name. Other large dumps, like those containing the abstracts of articles or those including only metadata about changes in the wiki (without the wikitext for each version) are also split in a similar fashion.

An important advantage of retrieving information from these dump files is that researchers have maximum flexibility as for the type and format of the information they want to obtain. As we will see in the next chapter, this also means that researchers can search for information about useful metadata present in the wiki text that was not originally included in the fields of the database. An example of this could be searching for special tags that mark quality content in Wikipedia, templates for assessing the status of articles (verifiability, disputed neutrality) or tracking the evolution of links and references inserted in an article over time.

Of course, whenever something is too good to be true there must be some hidden caveat. In this case, the downside is that one needs to use any software already available to import this information from the dump files, or write their own code to accomplish this (something that is not within the reach of many people). In the next chapter, we will review some existing tools that can help you in this endeavour. Here, we will describe in detail the content of some of the most popular dump files.

Finally, an important remark about the dump files is that every new file includes again all data already stored in prior versions, plus the new changes performed in the system since the last dump process, and excluding all information and metadata pertaining pages that have been deleted in that interval. In other words, we must understand these dump files as *snapshots* of the status of a Wikipedia language (or any other wiki of a Wikimedia project) at the time in which the dump process was triggered. Now and then, new people subscribing to the mailing list devoted to discuss the back up process, status and features of the dump files ¹³ asks about the existence of this kind of incremental dump feature. In case that we only had to deal with new changes, this could certainly be implemented without too much effort. However, the additional complication introduced by page deletions makes this way trickier to implement, and less useful (otherwise, the dump may show information about pages that do not exist in the wiki any more..., well, unless they are restored again in the future... funny enough, right?).

¹³<https://lists.wikimedia.org/mailman/listinfo/xmldatadumps-1>

1.7.1 Types of dump files

If you visit the Wikimedia Downloads center, and click on *database backup dumps* you will arrive at a page listing the progress of the process to generate all dump files for every Wikimedia wiki. Clicking on the link with the name of the project will lead you to the page summarizing all available dump files for it, along with the links to download them. The project code should be easy to interpret. Any link of the form *XXwiki*, where *XX* is a 2-letter (sometimes 3-letter) ISO code representing the language, identifies the Wikipedia in that language. Therefore, *eswiki* links to the dump files of Wikipedia in Spanish, *frwiki* to the dumps of the French Wikipedia, and so on.

As an example the filename *frwiki-20120601-pages-meta-history.xml.7z* tells us that the dump file belongs to the ensemble for the French Wikipedia, the date of creation of the file, the type (in this case, complete wiki text and metadata for every change in the wiki) the format (XML) and the compression algorithm (7-zip).

Table 1.1 summarizes many of the available dump files for the case of Wikipedia projects. You can refer to the MediaWiki DB schema ¹⁴ to learn more about the content included in each DB table.

Table 1.1: Summary of the different dump files available for any Wikipedia language

Type	Description	Format	Compression	Data and meta-data
Pages-meta-history	Complete wiki text and metadata for every change in the wiki.	XML	7-zip & bzip2	MediaWiki DB tables page, revision and text
Pages-logging	Administrative and maintenance tasks.	XML	gzip	MediaWiki DB table logging
Pages-meta-current	Only current (last) version of all pages.	XML	bz2	Page, revision and text tables in MediaWiki DB
Stub-meta-history	Metadata about changes in all pages.	XML	gzip	Page and revision tables in MediaWiki DB
Stub-meta-current	Metadata about last version of all pages	XML	gzip	Page and revision tables in MediaWiki DB
user_groups	List of users with special privileges	SQL	gzip	MediaWiki DB table user_groups
lang-links	Links to versions of a wiki page in other languages	SQL	7-zip	MediaWiki DB table langlinks
external-links	Links from a wiki page to other pages outside Wikipedia	SQL	7-zip	MediaWiki DB table externallinks

¹⁴http://www.mediawiki.org/wiki/Manual:Database_layout

category-links	Category tags inserted in a wiki page	SQL	7-zip	Mediwa Wiki DB table categorylinks
page-links	Links to other wiki pages in the same language	SQL	7-zip	Mediwa Wiki DB table pagelinks

1.7.2 Complete activity records: *stub-meta* and *pages-meta*

The two most popular types of dump files are *stub-meta-history* and *pages-meta-history*. These dump files contain information about every single change performed on any wiki page in a given Wikipedia language. In the Wikipedia terminology, these changes are called *revisions*. Thus, a revision is any change (creation or modification) that alters the content of a wiki page, producing a new version. The Program 1 caption shows an excerpt of the XML code store in one of these files.

These dumps (as well as other, such as *pages-logging* start with information about *namespaces*¹⁵, along with their names (that are frequently translated to the same local language). This information is a valuable aid to classify data pertaining to different namespaces, as well as to filter out data that is not relevant for the purposes of our study (for example, if we are only interested in articles or discussion pages).

After this, the file lists all revisions undertaken in every wiki page. The organization is shown in the Program 1 excerpt, with the metadata about the page (title, namespace, unique identifier and attributes) first, and then metadata and content for all revisions in that page. In this case, we find a revision that was carried out by an anonymous user, and thus only the IP address of the source of the incoming connection is stored.

In the Program 2 excerpt we see another two entries for a revision on the same page, this time undertaken by a registered user. In this case, we have the unique numerical identifier of the user in the system, along with her nickname.

The *stub-meta-history* dump contains exactly the same information with the sole exception of the whole wiki text for each revision, which is omitted. As a result, this can speed up significantly the parsing process to recover all information if we are only interested in metadata about pages and revisions. Nonetheless, in case that we do have any interest in tracing particular tags or types of content in the wiki text, we must use the complete history dump.

You can refer to the current version of the *page*¹⁶ and *revision*¹⁷ tables in the MediaWiki database to learn about the meaning of the fields included in these dumps. For many languages, the *rev_len* (length of revision in bytes) and the new *rev_sha1* field (with a hash of the text of every revision, using the SHA-1¹⁸ algorithm) are not available, yet. However, we will see that some data extraction tools can compute and store this information locally while parsing the dump file.

1.7.3 User groups

The *user_groups* dump contains information about special privileges assigned to certain users in the wiki. This is an exact dump (in SQL format) of the *user_groups* MediaWiki table¹⁹. In

¹⁵<http://en.wikipedia.org/wiki/Wikipedia:Namespace>

¹⁶http://www.mediawiki.org/wiki/Manual:Page_table

¹⁷http://www.mediawiki.org/wiki/Manual:Revision_table

¹⁸<http://en.wikipedia.org/wiki/SHA-1>

¹⁹http://www.mediawiki.org/wiki/Manual:User_groups_table

general, the number of different groups (or privilege levels) associated to users profiles will depend on the language. In big languages, we will find most (if not all) of the possible user levels currently considered in Wikipedia ²⁰. For other languages with smaller communities, some of these levels may be missing.

1.7.4 Logging dumps: administrative and maintenance tasks

Another interesting dump file that does not share the same fame and popularity as its revision history siblings is the *pages-logging* dump. In practice, this situation stems from the main goals of most data extraction tools for MediaWiki data, which were originally focused on extracting data for pages, revisions and text to build local replicas of a Wikipedia language. As a result, administrative actions were not as important, and for a long time they have remained quite unnoticed in Wikipedia research literature.

Today, next-generation data extraction tools such as WikiDAT (presented in the next chapter) solves this problem. The logging table in MediaWiki DB ²¹ contains unvaluable information about many different administrative and maintenance actions undertaken in a Wikipedia language. Gregor Martynus has recently created a draft list summarizing the different types of actions that we can find in this dump and their meaning ²². Some interesting examples of actions that we can trace in this dump are:

- *Deletions, moves or protection* of wiki pages.
- Registration of *new users* in the system.
- *Blocking* users, including the duration of the block.
- *Reviewing* actions performed in languages that has the *flagged revisions* extension enabled ²³.

The excerpts Program 3 and Program 4 show some example XML content of this dump for the case of the Simple English Wikipedia (*simplewiki*).

²⁰http://en.wikipedia.org/wiki/Wikipedia:User_access_levels

²¹http://www.mediawiki.org/wiki/Manual:Logging_table

²²<https://gist.github.com/2906718>

²³http://en.wikipedia.org/wiki/Wikipedia:Flagged_revisions

Program 1 Example of XML data stored in *pages-meta-history* dump

```
<mediawiki xmlns="http://www.mediawiki.org/xml/export-0.6/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mediawiki.org/xml/export-0.6/
http://www.mediawiki.org/xml/export-0.6.xsd"
version="0.6" xml:lang="fur">
  <siteinfo>
    <sitename>Vichipédie</sitename>
    <base>http://fur.wikipedia.org/wiki/Pagjine_princip%C3%A2l</base>
    <generator>MediaWiki 1.19wmfl</generator>
    <case>first-letter</case>
    <namespaces>
      <namespace key="-2" case="first-letter">Media</namespace>
      <namespace key="-1" case="first-letter">Speciâl</namespace>
      <namespace key="0" case="first-letter" />
      <namespace key="1" case="first-letter">Discussion</namespace>
      <namespace key="2" case="first-letter">Utent</namespace>
      <namespace key="3" case="first-letter">Discussion utent</namespace>
      <namespace key="4" case="first-letter">Vichipédie</namespace>
      <namespace key="5" case="first-letter">Discussion Vichipédie</namespace>
      <namespace key="6" case="first-letter">Figure</namespace>
      <namespace key="7" case="first-letter">Discussion figure</namespace>
      <namespace key="8" case="first-letter">MediaWiki</namespace>
      <namespace key="9" case="first-letter">Discussion MediaWiki</namespace>
      <namespace key="10" case="first-letter">Model</namespace>
      <namespace key="11" case="first-letter">Discussion model</namespace>
      <namespace key="12" case="first-letter">Jutori</namespace>
      <namespace key="13" case="first-letter">Discussion jutori</namespace>
      <namespace key="14" case="first-letter">Categorie</namespace>
      <namespace key="15" case="first-letter">Discussion categorie</namespace>
    </namespaces>
  </siteinfo>
  <page>
    <title>Pagjine principâl</title>
    <ns>0</ns>
    <id>1</id>
    <sha1 />
    <restrictions>edit=autoconfirmed:move=autoconfirmed</restrictions>
    <revision>
      <id>1</id>
      <timestamp>2005-01-25T06:55:26Z</timestamp>
      <contributor>
        <ip>24.251.243.233</ip>
      </contributor>
      <text xml:space="preserve">'''Benvignût al Vichipédie furlan!''''
[[Friûl]], [[Lenghe Furlane]], [[Culture Furlane]],
[[Statût regjonâl]], [[Europe]] [[Paîs Basc]]</text>
    </revision>
```

Program 2 Example of XML data stored in *pages-meta-history* dump (cont.). Lines have been formatted to fit the page width.

```
<revision>
  <id>55</id>
  <timestamp>2005-01-28T09:55:00Z</timestamp>
  <contributor>
    <username>Klenje</username>
    <id>1</id>
  </contributor>
  <minor />
  <text xml:space="preserve"> More wiki text
  here </text>
</revision>
<revision>
  <id>2456</id>
  <timestamp>2005-08-30T11:37:22Z</timestamp>
  <contributor>
    <username>CruccoBot</username>
    <id>29</id>
  </contributor>
  <minor />
  <comment>robot Adding: an, ar, bg, ca, co, da, de, en, es, fr,
id, it, ja, lb, mt, nl, nn, no, pl, pt, ro, ru, sc,
scn, simple, sv, tl, uk, vi, zh</comment>
  <text xml:space="preserve">{| width="100%" cellspacing="
0" cellpadding="5" style="
border:1px solid #996600;"
| bgcolor="#fff3f3" style="font: 95%
Verdana;"|
<big>Tu puedis lei articui in tantis lenghis
diferentis:</big>
{{Vichipedielenghis}}<div style="float: right;
" ><small>
[{{SERVER}}]{{localurl:Template:Vichipedielenghis|action=edit}}
Modificâ</small></div></td>
|}
{{Exzellent|14. Juni 2008|111222333}}
</text>
</revision>
</page>
</mediawiki>
```

Program 3 Example of XML data stored in *pages-logging* dump

```
<mediawiki xmlns="http://www.mediawiki.org/xml/export-0.6/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mediawiki.org/xml/export-0.6/
http://www.mediawiki.org/xml/export-0.6.xsd" version="0.6" xml:lang="en">
  <siteinfo>
    <sitename>Wikipedia</sitename>
    <base>http://simple.wikipedia.org/wiki/Main_Page</base>
    <generator>MediaWiki 1.20wmf4</generator>
    <case>first-letter</case>
    <namespaces>
      <namespace key="-2" case="first-letter">Media</namespace>
      <namespace key="-1" case="first-letter">Special</namespace>
      <namespace key="0" case="first-letter" />
      <namespace key="1" case="first-letter">Talk</namespace>
      <namespace key="2" case="first-letter">User</namespace>
      <namespace key="3" case="first-letter">User talk</namespace>
      <namespace key="4" case="first-letter">Wikipedia</namespace>
      <namespace key="5" case="first-letter">Wikipedia talk</namespace>
      <namespace key="6" case="first-letter">File</namespace>
      <namespace key="7" case="first-letter">File talk</namespace>
      <namespace key="8" case="first-letter">MediaWiki</namespace>
      <namespace key="9" case="first-letter">MediaWiki talk</namespace>
      <namespace key="10" case="first-letter">Template</namespace>
      <namespace key="11" case="first-letter">Template talk</namespace>
      <namespace key="12" case="first-letter">Help</namespace>
      <namespace key="13" case="first-letter">Help talk</namespace>
      <namespace key="14" case="first-letter">Category</namespace>
      <namespace key="15" case="first-letter">Category talk</namespace>
    </namespaces>
  </siteinfo>
  <logitem>
    <id>1</id>
    <timestamp>2004-12-23T07:28:37Z</timestamp>
    <contributor>
      <username>Netoholic</username>
      <id>515</id>
    </contributor>
    <comment>clearing MediaWiki namespace of legacy items - content was:
    '#redirect [[Template:1911]]'</comment>
    <type>delete</type>
    <action>delete</action>
    <logtitle>MediaWiki:1911</logtitle>
    <params xml:space="preserve" />
  </logitem>
```

Program 4 Example of XML data stored in *pages-logging* dump (cont.).

```
<logitem>
  <id>2</id>
  <timestamp>2004-12-23T07:30:05Z</timestamp>
  <contributor>
    <username>Netoholic</username>
    <id>515</id>
  </contributor>
  <comment>clearing MediaWiki namespace of legacy items - content was:
  '#redirect [[Wikipedia:MediaWiki namespace]]'</comment>
  <type>delete</type>
  <action>delete</action>
  <logtitle>MediaWiki:All messages</logtitle>
  <params xml:space="preserve" />
</logitem>
<logitem>
  <id>3</id>
  <timestamp>2004-12-23T07:30:18Z</timestamp>
  <contributor>
    <username>Netoholic</username>
    <id>515</id>
  </contributor>
  <comment>clearing MediaWiki namespace of legacy items - content was:
  '#redirect [[Wikipedia:MediaWiki namespace]]'</comment>
  <type>delete</type>
  <action>delete</action>
  <logtitle>MediaWiki:All system messages</logtitle>
  <params xml:space="preserve" />
</logitem>
<logitem>
  <id>4</id>
  <timestamp>2004-12-23T07:32:41Z</timestamp>
  <contributor>
    <username>Netoholic</username>
    <id>515</id>
  </contributor>
  <comment>clearing MediaWiki namespace of legacy items - content was:
  '#redirect [[Template:CompactTOC]]'</comment>
  <type>delete</type>
  <action>delete</action>
  <logtitle>MediaWiki:CompactTOC</logtitle>
  <params xml:space="preserve" />
</logitem>
</mediawiki>
```

Chapter 2

Data retrieval, preparation and storage

Now that we know more details about the information and metadata available from the different Wikipedia data sources, we turn to introduce some important aspects about the data retrieval process itself. After this, the next chapter will present some available tools to carry out this important task.

The main focus of this document are the Wikimedia dump files, and therefore we will cover some tips and useful recommendations that (hopefully) can save some time to researchers conducting Wikipedia data analysis with these files.

In the current version of the document, this chapter encompasses the most critical pieces of advice and recommendations that I could think of regarding Wikipedia data extraction. Thus, it is still far from complete, and it will be expanded as I find more free slot to expand this information with more tips and further assessments.

2.1 RSS to notify updates

A very usefull service that is still overlooked by most Wikipedia researchers today is the utilization of RSS notification channels to check for new updates of the dump files that they regularly use.

For every Wikipedia language, a special page on the Wikimedia Downloads site lists the latest correct versions of all dump files produced for that language. The structure of the URLs to reach these pages is always the same:

```
http://dumps.wikimedia.org/project-name/latest/
```

Where we must place instead of *project-name* the name of the Wikipedia language we want to track (frwiki, dewiki, enwiki, eswiki, etc.). Now, if we inspect the content of this page, we can quickly notice some links containing the tag *rss* in their URL, like:

```
frwiki-latest-pages-meta-history.xml.7z-rss.xml
```

In fact, this is the link to a RSS syndication channel to which we can subscribe to be notified about new versions of this dump file (in this case, *pages- meta-history* for the French Wikipedia). Then, we can follow this channel with our favourite aggregator to get this notifications or, even smarter, use this channels to automatically trigger some piece of software that retrieves the new dump file to a local machine, and load the new information in a fresh database copy.

2.2 The particular case of the English Wikipedia

The English Wikipedia is always a particular case not only for the data extraction process perspective, but for many other macroscopic studies that we were willing to implement. As of May 2012, the complete *pages-meta-history* file of *enwiki* stores text and metadata for 444,946,704 revisions performed in 27,023,430 pages (in all namespaces, not only for the main namespace of articles).

With such scores in mind, it quickly becomes apparent that researchers must think very carefully not only *what* kind of analysis they want to carry out but also *how* they plan to implement it. In practice, this is not quite a serious issue for other smaller languages, even if they are in the list of the largest Wikipedias. Nevertheless, it is very probable that if an analysis takes several days to complete in *eswiki* or *frwiki*, it will take *weeks* (or even a whole month) to complete for the English Wikipedia.

This is the kind of situations that we must try to avoid, as much as possible. As an example, in the old days (where “old” is 2008) where the dump process for the English Wikipedia could take several weeks to finish, it was quite frequent that any problems (in the form of a power outage, an error in code, problems in database connectivity, or many other impoderables) prevented this process from finishing correctly. As a consequence, researchers had to wait sometimes more than one year to have new, updated dump files.

Fortunately, today the situation is very different, but researchers should take this lesson with themselves to overcome the same problems in their own processes. As a rule of thumb, if the execution time of your analysis is dangerously approaching the 7 days threshold, you should redesign the implementation details. Otherwise, you would be leaving too a large room for Murphy’s law to smack down your process, and force you to start it all over again.

2.3 Computing extra metadata

In the previous chapter we touched upon the contents included in the dump files with the complete history of changes for each Wikipedia language. As we mentioned, one of the advantages of this approach is that, in the *pages-meta-history* dump we have the whole wiki text associated to every revision stored in the file. This opens up an endless number of possibilities for researchers interested in tracking changes in particular parts of the page content, or the introduction of specific templates and tags adopted by the Wikipedia community to flag certain conditions about the page status or its quality.

A straightforward example is to track the introduction of the tag to label an article as a *featured article*, that is, an article of very high quality in that language¹. For this purpose, the usual approach is to include some extra logic in the software implementing the data parsing of XML dump files to look for these special tags in the text of the revision. One of the possible ways to accomplish this is to use regular expression², a flexible tool (but also somewhat complex and prone to coding errors) that is present in popular programming languages such as Python, Perl or Ruby.

However, a word of caution is in place at this point. In the first place, while there are some techniques that can let us speed up the preparation of the regular expression to make the pattern matching process faster (like the use of *re.compile* in Python), this process may take some time. This is true for revisions with very large text chunks and also for complex patterns. For example, bear in mind that the template syntax for marking featured articles

¹http://en.wikipedia.org/wiki/Wikipedia:Featured_articles

²http://en.wikipedia.org/wiki/Regular_expression

in languages other than English can be translated into the local language, and you can find yourself composing patterns to match these tags in Hebrew, Farsi or Russian. For this, the use of UTF-8 codification can save us from getting a terrible regular expression headache. In the parser code included in WikiDAT[[TODO:ref!]] you can find examples of the patterns use to match the featured article tag in 39 Wikipedia languages, for a recent study that I have implemented with other colleagues and it is currently under review.

In conclusion, my point here is: you can use certain tools (like regular expressions) to look for interesting patterns in the text of revisions. But you must always remember that this comes with a high price: computational time. In particular, if you are thinking about implementing some type of fine grain NLP analysis, for example to apply sentimental analysis, on the wiki text of all revisions of the English Wikipedia, then you should certainly think about distributed computing alternatives (where terms like cluster, Hadoop, map-reduce, NoSQL and other buddies come into place).

All the same, if you are just interested in tracking a small subset of tags like in our featured articles example, but also for good articles, pages with disputed neutrality, introduction of templates to call for more references, and many other cases, the pattern-matching approach may well fit your needs in a reasonable execution time.

2.4 Practical tips and assessment

The aim of this last section is to compile useful advices that you can apply to make your life easier analyzing Wikipedia data. As I said in the presentation of this chapter, this section should expand in future versions of this document to bring more pragmatic recommendations.

2.4.1 DRY and ARW

A couple of simple tips that we all (even myself) have broken (several times). DRY stands for *Don't Repeat Yourself* (some people add a nice appeal to the end), whereas ARW accounts for *Avoid Reinventing the Wheel*. These are two important premises sustaining effective software development. They translate into two important messages:

- If you find that you are writing code that will solve *this* problem but not *any similar* problem that you may find in the future, then think it twice before going ahead with your typing. It is difficult to write reusable code but it will save you precious time later, so it worths the effort.
- Before trying to code your own super-duper-awesome solution for a given problem, look around (with your favourite search engine and exploring well-known public code repositories like SourceForce, Github, BitBucket and so forth) to see if anybody else already attempted to solve the same problem. An interesting property of software engineering (stated by Fred Brooks a long time ago[[TODO:ref!]]) is that, whenever you face a new problem, you should prepare yourself to throw your first implementation to trash. This has nothing to do with your personal skills, but with the inherent nature of problems in software engineering and how we think about the problem different once we have really understood all the low level details and caveats that were hidden at first sight.

2.4.2 Data storage: hardware

As I have explained before talking about extra metadata that we can retrieve from the text of revisions, there are two possible alternatives that we can follow for storing our data locally:

- Using a single machine (as powerful as we can afford) to undertake the data preparation and analysis.
- Resorting to use a cluster of machines, applying modern techniques for process parallelization such as map-reduce, or using cutting-edge distributed database engines following the NoSQL paradigm (like MongoDB or Cassandra).

My personal recommendation regarding this is to keep using a single machine unless you are absolutely sure that your problem will take a disproportionate amount of time to complete. It is true that, today, technical (and business) interests are determined to present new technologies like map-reduce or NoSQL as the answer for all our problems. However, anyone who has been involved in algorithm parallelization for some time can tell you that this can be all but a trivial task. Moreover, it is definitely out of the reach for many researchers without a strong background on programming, systems architecture and parallelization, and having strong skills using the software packages implied in this approach.

At the same time, some new hardware technologies can facilitate (to some extent point) our life in single-machine analysis. Multi-core CPUs are now quite affordable, so provided that we can buy a reasonable amount of RAM for our system (as much as we can afford) we can execute multiple processes for different Wikipedia languages at the same time. We will also be able to take advantage of the multiple-chunks dump format in the English Wikipedia, parsing several chunks at the same time. With this approach, the last version of WikiDAT (at the time of writing) can parse the whole English Wikipedia with 6 concurrent subprocesses targeting between 26 and 28 chunks each in about 44 hours. And in that time, it also calculates some extra metadata such as SHA-256 hashes for each revision, the length of the text, and presence/absence of featured article tags in the wiki text.

System memory (RAM) is another key factor with an impact on performance, specially for the data analysis portion. Loading data into a local database (or any other storage infrastructure) does not typically require substantial amounts of memory to execute properly. In fact, the current procedure is usually to recover the data directly from stream created by the compression software while inflating the original data file, capture the relevant information to be stored, and discard the rest. As a result, the *memory footprint* of this programs should not be very high. However, when we get to the analysis part, there can be quite a large difference between having the indexes of your huge table loaded in memory to look for specific entries in a database or not. This difference can be in the order of hours or even days.

Solid state drives are the new fastest gunmen in storage village. Many manufacturers have been able to rectify the initial problems related to unreliable writing performance in their first models, and they are now producing pieces of finest hardware suitable for serious, professional applications requiring maximum speed in data transfer at an affordable cost (well, at least while we keep below the 300 GB threshold, at the time of writing). An important property of SSD technology is that, compared to their close relatives based on magnetic technology, SSDs can be several orders of magnitude faster (and now, without noticeably degrading their writing performance). Yet another advantage of SSDs (specially for GNU/Linux users) is that we can configure one of these devices to act as a *swap*³ device, expanding the amount of (virtual) memory granted to the system. In practice, since the read/write speed of many of these disks is now quite fast, what we obtain is a system with much more memory without the downsides of the very slow magnetic technology in traditional drives.

Table 2.1 summarizes the main hardware factors that we have commented so far, along with some recommendations.

³http://en.wikipedia.org/wiki/Swap_partition

Table 2.1: Key hardware factors that affect performance in data extraction and analysis

Factor	Influence	Recommendations	Caveats
CPU	Execution speed of instructions	Multi-core processors allow us to run several concurrent processes to extract data and analyse different languages. Faster CPUs (higher frequency) will cut down execution time of our programs and database operations.	Limited number of CPUs/cores in a single machine (currently 8-16)
RAM memory	Speed of calculations, size of data sets	More RAM will let us load larger data sets or database tables in memory, reducing computation time.	RAM chips can be expensive (ratio cost/bit). Limited size of memory allowed in some systems (except for professional servers).
SSDs	Execution speed of I/O bounded tasks	Solid-state drive technology is now mature enough for serious applications. Good trade-off solution (ratio cost/bit). Orders of magnitude faster than traditional magnetic drives. We can reserve part of the drive for virtual memory, effectively expanding system memory with little impact on performance.	Beyond certain capacity (circa 250 GB) SSD devices are still somewhat expensive.
Hybrid hard drives	Large storage capacity, faster data transfers	HHDs are another compromise solution, integrating a large magnetic disk with a small SSD acting as a local cache in the same device. Large storage capacity with faster data transfer. Mandatory for working with very large data sets (like whole wiki text in a Wikipedia language)	Not valid for some applications as for SSDs, only required for very large storage capacity

On-site clustering	Parallelization of very complex and time-consuming tasks	A local cluster connected to a reliable, fast communication network (Gigabit Ethernet or Fibre Channel) is the only way to accomplish very complex analysis involving difficult calculations. Examples are NLP of a whole Wikipedia language, authorship at the word level, and temporal evolution of social networks.	Significant effort to set up such an infrastructure in a proper way. Required skills with associated technologies and physical space for installation. It can be quite expensive.
Clustering in the cloud	Same as before with reduced cost	Cloud services such as Amazon EC2 offer access to scalable computational clustering at an affordable cost. Users only need to deal with software configuration, data load and execution of analyses. Users are charged for execution time.	Software, libraries and data must be loaded and configured every time we run a new analysis, so these tasks should be automated. May become expensive in the long-term for conducting many analyses.

2.4.3 Operating system and file system support

Depending on the scale of your analysis (that is, whether you work at the macro level with aggregate scores or at the microscopic level, with high-resolution data), you may find some surprises while preparing for your data analysis. For macroscopic studies, one usually work with tables containing metrics and aggregated scores that have been computed in the data preparation phase, so that your tables and data source files are not going to be very large. In fact, it is perfectly possible to run macroscopic analyses even for the English Wikipedia in a modern, multi-core laptop with 4GB of RAM and an SSD, with good performance.

However, high-resolution analysis is another story. If you want to apply NLP techniques over the whole set of words in the wiki text of a certain Wikipedia language, you will probably have to work with the decompressed data file, which can take several TB of storage space (specially for the largest languages). The same stands, for example, for analyzing the changes in the content of articles, computing authorship information and tracing these changes over time for the whole set of encyclopedic entries in a given language.

In all these cases in which there are clear reasons to think that we must work with files consuming TB of storage space, we must think carefully about the type and configuration of the system in which we want to run our analysis. Leaving aside clustering solutions, it is possible to configure a server infrastructure with a storage capacity spanning more than 2TB, but we need to follow some recommendations:

- *64-bit platforms*: It is mandatory for high-resolution analysis to run on 64-bit platforms, prepared to handle large data structures both in memory and in their file system.
- *File system*: Another important choice is to use a modern filesystem that supports a very

large file size⁴. For example, in GNU/Linux *ext4* or *xf*s are good options. You should also pay attention to the size of your partitions, to use a technology accepting the configuration of partitions with a size larger than 2TB. There are several options available, including the use of GPT⁵ partition tables, configuration of multiple RAID devices and joining several standard partitions of up to 2TB together using LVM⁶.

Finally, regarding the implementation of your storage infrastructure, in GNU/Linux the performance offered by software RAID configurations (using *mdadm*) is quite decent for many applications in data analysis, in particular if we can use SSD devices. However, we must remember that software RAID consumes part of the CPU resources that will not be available for our processes. Hence, my recommendation is always to acquire even a entry-level hardware RAID card from a major brand in the business (such as Adaptec) to get better performance.

2.4.4 Data storage: database engines

Over the past 5 years, MySQL has been the database engine that I have used consistently to conduct Wikipedia data analysis. Admittedly, this is strongly linked to the fact that I have been interested in studies at the macroscopic level. For high-resolution analyses such as working with the whole set of wiki text in a Wikipedia language, modern NoSQL alternatives like MongoDB or Cassandra can be a better choice. As for relational databases, PostgreSQL is the other obvious choice, but practice have taught me that getting to know your database engine extremely well can be the difference between spending several days or few minutes to complete a query. Therefore, this section is focused on MySQL.

Many people become quite surprise whenever I mention that, running on MySQL, I have had no major issues running Wikipedia data analysis for multiple languages, even for the English Wikipedia (now with nearly 450 million revisions). Likewise, my colleague Antonio Reinoso has been able to process nearly 1 billion entries with information about Wikipedia traffic (for a single month) on MySQL without a problem. The key point here is: default MySQL server configuration is useless for any serious application (not only for Wikipedia data analysis). Thus, provided that you configure your sever in a proper way, things will change dramatically.

The bad news is that configuring MySQL (as well as any other database engine) is not a easy work. In fact, another complication is that the optimal configuration for one machine can be perfectly useless in a different computer. The net result is that there is no magic configuration (or “silver bullet”) that can be shared accross multiple servers. In fact, MySQL configuration is a multifaceted issue that deserves a whole book on its own [[TODO:Ref High perf. MySQL 3rd edition]]. I cannot stress the importance of getting a deep knowledge of your database engine if you want to conduct serious data analysis.

Another curious aspect is that, so far, for most of my applications I have been using the MyISAM eninge, which has now become deprecated. Its replacement, InnoDB, is without a doubt a more advanced engine for many business applications. However, for the specific needs of data analysis these advantages do not stand out so clearly. For example, in data analysis we frequently create data tables firsts, and we only perform read operations thereafter. Concurrent updates (writings), which are common in business applications, are very scarce in data analysis. Moreover, counting queries usually perform much faster in MyISAM than in InnoDB. Finally, InnoDB presents a much longer list of possible parameters for server fine-tuning that we need to carefully study and test.

⁴http://en.wikipedia.org/wiki/Comparison_of_file_systems

⁵http://en.wikipedia.org/wiki/GUID_Partition_Table

⁶http://en.wikipedia.org/wiki/Logical_volume_management

In conclusion, I offer in Table 2.2 a list of some of the most important parameters to which we must pay attention in MyISAM. Nevertheless, since InnoDB is the future this section will be expanded in later versions of this document to include useful tips for InnoDB. You can check the whole list of variables and their meaning on the online manuals for MyISAM and InnoDB. I also strongly recommend the very good book by Schwartz et al. [\[TODO: REF!!\]](#) to get additional insights and many useful tips to configure your server.

Table 2.2: MyISAM server variables with high influence in performance of the data extraction and analysis process

Variable	Purpose	Recommendations
key_buffer	Memory space to load keys that speed up search and ordering operations	Make it as large as possible without consuming more than 50% of total available memory if you run the analysis (Python, R) in the same machine, or 75% in a dedicated database server.
max_allowed_packet and bulk_insert_buffer_size	Control the size of packets for information insertion in database	Make them large enough to avoid hindering the data insertion. 256M or 512M is usually enough.
sort_buffer_size	Size of the buffer used to sort out values in queries	In data analysis we usually sort result values using ORDER BY. If you can allocate some memory (1 or 2 GB) for this purpose, it can speed up this type of queries.
myisam_sort_buffer_size	Besides the previous one, MyISAM has its own buffer to sort results	Same as before
read_buffer_size and read_rnd_buffer_size	Control the size of buffers used to load data from disk	Larger buffers will let us you to read data from disk faster.

Before closing this section (and the chapter) I would like to mention two important features of InnoDB that do have an impact in data analysis. The first aspect is that, by default, InnoDB stores all data (and keys to speed up search operations) in a single file that can grow up to a huge size. Besides, in case that you delete some of the tables, or entire databases, the space consumed by this file is never claimed back, preventing us to free storage capacity in our system. This can be a problem in platforms with high computational power but with limited storage capacity. The solution is to configure the `innodb_file_per_table` server variable ⁷. At least, in this way we can perform some operations to free up some storage space.

The second aspect is that we can set up different *tablespaces* to place different tables in different directories ⁸ (maybe in different devices, such as distinct SSDs). This can be an important factor to speed up the analysis, running different processes that target their own table in different devices. This is a kind of low-level process parallelization without entering in the details of distributing different tasks in a high-level programming environment. Using table

⁷<http://dev.mysql.com/doc/refman/5.0/en/innodb-multiple-tablespaces.html>

⁸<http://dev.mysql.com/doc/refman/5.1/en/partitioning-overview.html>

partitioning, it is important to understand the nature of the operations that we are going to perform to get the maximum benefit from this maneuver. For example, if we are carrying out a longitudinal analysis by month, it will make sense to partition data among different tables for each month. If our analysis goes per page or per user, we can also partition our tables accordingly.

Chapter 3

Available tools for Wikipedia data extraction and analysis

3.1 Here be dragons

Reviewing previous research literature conducting quantitative analyses with Wikipedia data can be a daunting task. Recent work in progress has reported nearly 2,000 works only for peer-reviewed venues and scientific publications ¹. However, this bright beam research work turns into darkness (or at least twilight) when we speak about publication of source code to undertake these analyses. I'm not going to mention here all possible examples, but we have several cases of stunning applications like HistoryFlow (developed by F. Viégas and Martin Wattenberg while working for IBM, and frequently cited in Wikipedia research) or WikiDashboard ², for which their code is not publicly available as open source.

In practice (and I do share my own quota of blame here), researchers are too busy to devote the necessary time to publish the code and data sets associated to their studies and research reports, as to facilitate the labour of other colleagues willing to follow their footsteps and expanding their line of work. The most important factor for this lack of interest is the strong focus on reviewing and polishing research documents (reports, journal and conference articles, etc.) in the scientific community (at least in Computer Science). This relegates data sets, tools and, in general, study replicability to an obscure corner in the background.

In spite of this, programming languages and modern statistical environments such as R (that we will visit in the next part) offer built-in support for creating modular applications and libraries that can be easily loaded and integrated in new software. Therefore, it is not the lack of technical support, but the absence of a clear interest from the scientific community so far in study replicability which stops us from mapping more unexplored territories in detail for other to enter new research regions.

That said, in Wikipedia research we do not have a hopeless panorama, since there are some open source tools and suites that can save us time implementing our own studies. In this chapter, we briefly present some of these open source tools, along with the pointers for readers interesting in learning further details about them.

¹http://wikilit.referata.com/wiki/Main_Page

²<http://wikidashboard.appspot.com/>

3.2 Wikistats and report cards

Wikistats ³ is a website presenting multiple statistics about the overall evolution of all Wikimedia projects. The analysis is powered by a set of Perl scripts developed by Erik Zachte, data analyst at Wikimedia Foundation. The code of this scripts is available for download from the same website This site was the first to provide a comprehensive overview of all Wikimedia projects, and it has evolved over time to improve the display of statistical results in a more meaningful way. From Wikistats, visitors can also download CSV files with all data generated by the extraction scripts, ready for use.

In this regard, a new project has been launched by Wikimedia Foundation, with the backup of the new Wikimedia Labs branch, to offer monthly report cards that present similar information in a more interactive and appealing fashion, using new technologies such as JavaScript and HTML5 ⁴. According to the information on that page, the source code for creating these report cards, Limn ⁵ will be available very soon, under an open source license (as usual with all projects developed by WMF).

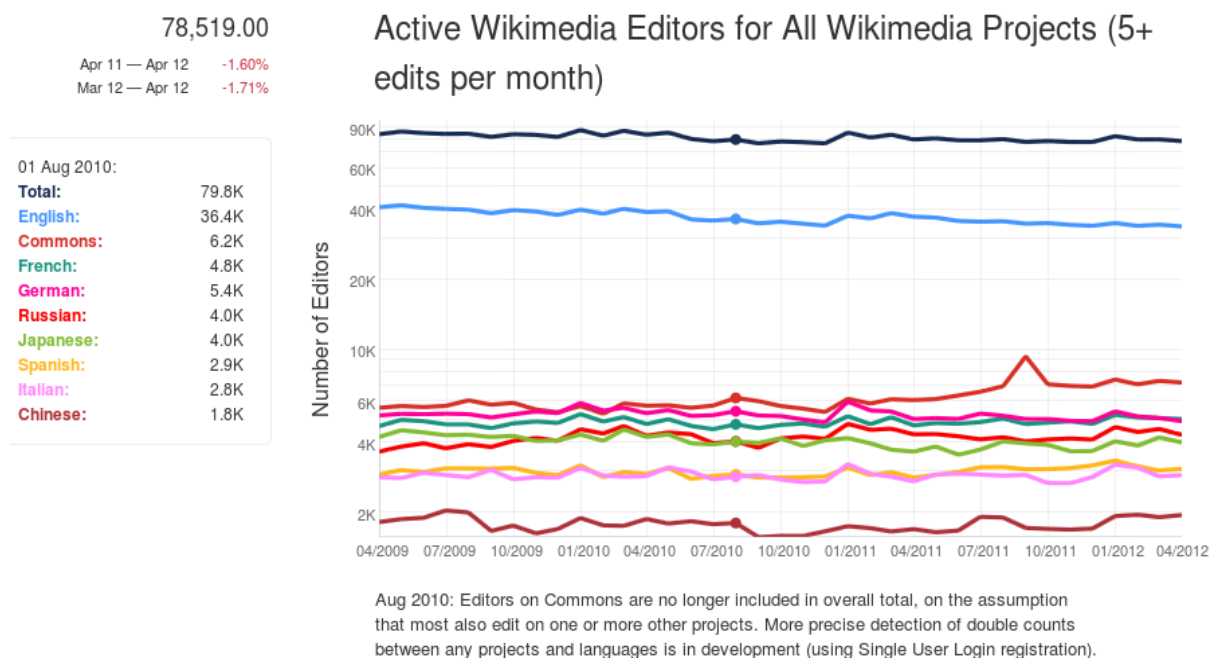


Figure 3.1: Example of the new reportcards produced by WMF.

3.3 StatMediaWiki and Wikievidens

StatMediaWiki ⁶ is another tool to calculate metrics and statistics about the evolution of any MediaWiki-powered site, created by Emilio José Rodríguez (*Emijrp* in Wikipedia). It also exports this pre-processed information for later consumption, either as HTML pages or CSV data files. The code is licensed under GNU GPLv3.

³<http://stats.wikimedia.org/index.html#fragment-14>

⁴<http://reportcard.wmflabs.org/>

⁵<https://github.com/wikimedia/limn>

⁶http://statmediawiki.forja.rediris.es/index_en.html



 Universidad de Cádiz
 Oficina de Software Libre

Language
 Español
 English

Main menu

- Main page
- News
- Demo
- Manual
- Development
- Source code
- Papers
- Contact

Welcome to StatMediaWiki home site

StatMediaWiki is a project that aims to create a tool to collect and aggregate information available in a MediaWiki installation.

Results are static HTML pages including tables and graphics that can help to analyze the wiki status and development, or a CSV file for custom processing.

StatMediaWiki is free software under the GPL v3 or higher license. Some functionalities in StatMediaWiki are adapted from those in [StatsVN](#).

[<< EvalMediaWiki](#)
[WikiRing](#)
[WikiEvidens >>](#)

Libre Software and Open Knowledge Office, Universidad de Cádiz





Content of this site is licensed under a [Creative Commons Licence](#).

Figure 3.2: Project page for statMediaWiki at RedIRIS forge.

Wikievidens⁷ can be interpreted as the evolution of the previous tool, intended to provide a statistical and visualization software for wikis. It is still on alpha stage.

3.4 Interacting with the MediaWiki API

Several libraries and applications have been developed to interact with the MediaWiki API to automate certain tasks in Wikipedia, such as developing bots to perform regular maintenance duties. These tools can also be very useful for researchers, since they provide convenient wrappers to access the API within popular programming languages like Python.

3.4.1 Pywikipediabot

Pywikipediabot⁸ is a Python library created to facilitate the development of bots for Wikipedia. As such, it is ideal to access the functionalities of the MediaWiki API for any Wikipedia language from Python. It is also very easy to learn, and there are numerous examples that can be found of existing bots to learn the code.

However, you should take into account the usual recommendations as for accessing the Wikimedia API at a very fast pace, since you would run the risk of being banned by system

⁷<http://code.google.com/p/wikievidens/>

⁸<http://www.mediawiki.org/wiki/Pywikipediabot>

Project Information

 Recommend this on Google

[Project feeds](#)


Code license
[GNU GPL v3](#)

Content license
[Creative Commons 3.0 BY-SA](#)

Labels
wiki, wikipedia, mediawiki, visualization, statistics, academic, interactive, downloader, processing, nlp, socialnetworks, graphviz, pylab, matplotlib, numpy

Members
[emi...@gmail.com](#)

Featured

 [Wiki pages](#)
[Contribute](#)
[Features](#)
[FirstSteps](#)
[Install](#)
[Show all »](#)

Links

Groups
[WikiEvidens](#)

What is WikiEvidens?

WikiEvidens is a statistical and visualization tool for wikis. It is developed by [Emilio J. Rodriguez-Posada](#) and it is on alpha stage.

Features

- Wiki datasets **downloader**
- Wiki XML **preprocessor**
- Global, page-by-page, user-by-user and sample **analysis**
 - Summary
 - Activity by year, month, day of week, hour
 - Networks visualization
 - Authorship scanner
 - Many more very soon...
- **Export** in several formats

Documentation

Learn to install and use WikiEvidens. We are working in a complete [Documentation](#).

Contribute

If you want to send any **suggestion**, reach me at [emijrp@gmail.com](#). For **bugs**, you can use the [issues](#) section.

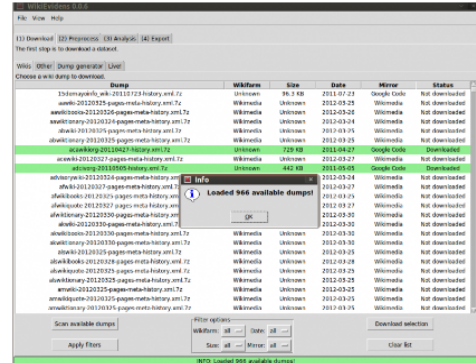


Figure 3.3: Project page for wikievidens at Google Code.

administrators. Only bots that have been officially approved and get that status in the system can perform such actions at full speed.

3.4.2 Python-wikitoools

A second option for accessing the MediaWiki API from Python is python-wikitoools⁹. The tool can be downloaded from the page project hosted at Google Code, or alternatively through Pypi, the official Python packages repository.

3.4.3 Mwclient

Finally, you can also check mwclient¹⁰, a framework for accessing the MediaWiki API in Python written by Bryan Tong Minh for personal use in his own bots. According to the information from the README file in the latest version available (0.6.5):

Mwclient is a client to the MediaWiki API <<http://mediawiki.org/wiki/API>> and allows access to almost all implemented API functions. Mwclient requires Python 2.4. This version supports MediaWiki 1.11 and above. However, for functions not available in the current MediaWiki, a MediaWikiVersionError is raised.

⁹<http://code.google.com/p/python-wikitoools/>

¹⁰<http://sourceforge.net/projects/mwclient/>

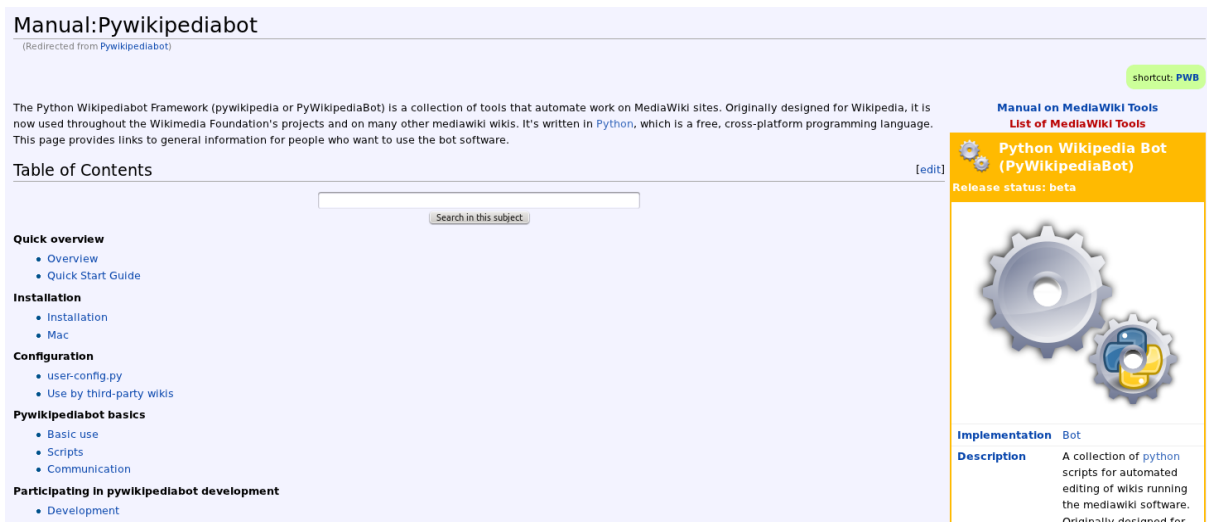


Figure 3.4: Snapshot of the Pywikipediabot page at mediawiki.org

3.5 WikiTrust (for data analysis)

WikiTrust¹¹ is an open-source, on-line system to calculate reputation metrics for Wikipedia authors and content. WikiTrust is hosted by the Institute for Scalable Scientific Data Management at the School of Engineering of the University of California, Santa Cruz, and their main developers are Lucal de Alfaro, Bo Adler and Ian Pye. WikiTrust has been one of the most popular research tools for calculating metrics about Wikipedia authoring, including several research papers that can be accessed from the same website. It has also been applied to the study of vandalism and reverted content in Wikipedia, since along the process the software tracks the changes or moves experimented by any word inside a Wikipedia article (as well as the author of the revision performing those changes).

This authorship and reputation information is available online for being displayed on any Wikipedia article thanks to an add-on plug-in developed for Mozilla Firefox¹². A set of 3 different APIs¹³ is available for accessing the information computed by WikiTrust. The project authors are actively looking for support to provide these data for Wikipedia languages other than English, after a power outage that brought down all international WikiTrust support once available.

The code is released under a BSD-like open source license. In addition, many files implementing the core algorithm are written in Ocaml, a programming language suitable for distributed processing in computer clusters. Since computing this algorithm for an entire Wikipedia language is a daunting task, it must be accomplished by this kind of parallel computing infrastructure. Even then, it can take a substantial period of time to complete (nearly a month for the whole English Wikipedia, according to the last comments from authors). This also depends on the number of nodes used for the calculation.

¹¹<http://www.wikitrust.net/>

¹²<https://addons.mozilla.org/en-US/firefox/addon/wikitrust/>

¹³<http://www.wikitrust.net/vandalism-api>



python-wikitools

Python scripts and modules to interact with the MediaWiki API and source code for some en.wikipedia bots

Project Home Downloads Wiki Issues Source

Summary People

Project Information

+5 Recommend this on Google

[Project feeds](#)

Code license
[GNU GPL v3](#)

Content license
[Creative Commons 3.0 BY-SA](#)

Labels
MediaWiki, Wikipedia, Python

Members
[MrZmanwiki@gmail.com](#)
[2 committers](#)

Featured

Downloads
[wikitools-1.1.1-1.noarch.rpm](#)
[wikitools-1.1.1-1.src.rpm](#)
[wikitools-1.1.1.tar.gz](#)
[wikitools-1.1.1.win32.exe](#)
[wikitools-1.1.1.zip](#)
[Show all »](#)

Python package to interact with the MediaWiki API. The package contains general tools for working with wikis, pages, and users on the wiki and retrieving data from the MediaWiki API. There is also the source for some en.wikipedia specific scripts using the framework, including the source for [Mr.Z-bot @ en.wikipedia](#).

- The wikitoools module requires Bob Ippolito's [simplejson](#) module or the json module in Python 2.6+.
- Chris AtLee's [poster](#) package is needed for file upload support in all Python versions, but it is not required for use. If not present, file upload support will be disabled.
- Version 1.1.1 was released on 14 April 2010. Source downloads with a setup.py script, a Windows installer, and an RPM are available in the Downloads tab or the right sidebar.
- wikitoools will be roughly following the MediaWiki release cycle for major releases, ensuring that each release is compatible with the version of MediaWiki released at the same time. If you are using the development alpha version of MediaWiki (as Wikipedia does), you should consider using the development version of wikitoools, which can be downloaded via SVN here (from the "source" tab). Old versions can be downloaded from the [deprecated download list](#).
- Note that due to developer time constraints, only the most recent release and the SVN version are supported and receive bugfix releases. For major bugs affecting old versions, a patch may be released.
- Documentation is available on the [wiki](#)
- Some bot scripts (not the framework itself) require the [MySQLdb module](#) and a MySQL server.
- Scripts in the "pywiki" branch directory require [Pywikipedia](#). Note that these scripts are unmaintained and may no longer work.
- If you'd like to help out with this, send me an email (mrzmanwiki {at} gmail {dot} com) or leave a message on my en.wikipedia user talk page.
- [README](#)
- Also available on the Python package index: [pypi](#)
- Author: [Mr.Z-man](#) @ en.wikipedia
- Some assistance and code by [bjweeks](#)

Figure 3.5: Snapshot of the python-wikitoools at Google Code.

3.6 Pymwdat

We can find traces of the first attempts to create applications to automate the analysis of Wikipedia data back to 2008. Dmitry Chichkov starts to develop wredese, a tool to support the analysis of reverts and vandalism actions in Wikipedia. Later on, this project became pymwdat¹⁴, a more general framework to implement these analyses, as well as the creation of overall statistics about any Wikipedia language.

According to the online documentation for this project:

Requirements:

- * Python 2.6+, Linux;
- * OrderedDict (available in Python 2.7 or <http://pypi.python.org/pypi/ordereddict/>)
- * 7-Zip (command line 7za)

Input Alternatives:

- * Wikipedia history dump (e.g. enwiki-20100130-pages-meta-history.xml.7z);
- * Wikipedia page(s) name / category (use grab-pages.py);
- * Wikipedia page(s) history (.xml/.7z/.bz2/.gz).

Input Processing:

- * Detects reverts, reverted edits, revert wars, self-reverts;
- * Filtering, labeled revisions data sets management, etc;
- * Calculates users/page counters, revert rates, page diffs, etc

¹⁴<http://code.google.com/p/pymwdat/>



Figure 3.6: Snapshot of the WikiTrust Firefox add-on in action, colouring Wikipedia content according to the reputation metrics computed by the tool.

(~1 week/C2Duo/Full English Wikipedia dump).

3.7 Wikimedia-utilities

Wikimedia-utilities¹⁵ is a very interesting piece of software that have not received general attention yet. This software has been created by Aaron Halfaker, from Univ. of Minnesota, while working in a Summer internship at Wikimedia Foundation premises in San Francisco. The software is written in Python, and it has been conceived as a general-purpose application that distributes the analysis of Wikipedia data dumps over different subprocesses running on a multi-core machine.

The main advantage of this application is that it is extensible, namely that we can extend the code to include additional calculations or any other tasks that we want to perform on pages, revisions or their associated wiki text as the dump file is decompressed. As an example, this baseline code has inspired the design of a new tool for tracking reverts of article content in Wikipedia accurately, written by Fabian Flöck¹⁶.

3.8 WikiDAT: Wikipedia Data Analysis Toolkit

In April 2009, I released WikiXRay¹⁷, a Python tool to automate the analyses included in my PhD. dissertation, comparing the 10 largest Wikipedia languages at that time (by the official article count) from different quantitative perspectives[[TODO:CITATION!!]]. The main advantage over other similar tools is that WikiXRay also presents R code implementing the actual analysis part, and not only the data extraction process (which was written in Python). So, this is somehow one of the first available software for Wikipedia data analysis that really made an

¹⁵<https://bitbucket.org/halfak/wikimedia-utilities>

¹⁶<http://people.aifb.kit.edu/ffl/reverts/>

¹⁷<http://meta.wikimedia.org/wiki/WikiXRay>

effort to open up the data analysis part for other researchers to check the code and reuse it for their own purposes.

However, advances in available libraries for XML parsing, and new proofs of concept such as *wikimedia-utilities* has made me think that it could be a good idea to try to rewrite the old code, creating a new tool that I have called WikiDAT [[TODO:Citation]]. At the time of writing, it is still in experimental phase as I am able to find free slots to clean up and organize the multiple (and I have really many of them) code snippets that I have created for undertaking past analyses with Wikipedia data.

At the time of writing (end of June, 2012) WikiDAT is shipped with new versions of parsers for the *pages-meta-history* and *pages-logging* dump files. Regarding the pages meta history, Tables[[TODO:internal-ref]] summarizes the data fields that are currently extracted to a local MySQL database, organized in 5 different tables:

- Table *page* stores information about all pages in a Wikipedia language.
- Table *revision* contains metadata about all revisions performed in a Wikipedia language.
- Table *revision_hash* stores the id of every revision as well as an SHA-256 hash of the wiki text associated to it.
- Table *people* lists the numerical identifier and nickname of all registered users in a Wikipedia language.
- Table *logging* with information about administrative and maintenance tasks undertaken in a Wikipedia language.

The original definition of these tables is based on the tables of the same name defined in MediaWiki, except for table *people* which is new. Fields in bold characters identify keys to speed up search and sorting operations in database queries.

Table 3.1: Fields and values included in table *page* as defined in WikiDAT

Field name	Possible values	Description
page_id	Positive integer > 0	Unique numerical id of the page
page_namespace	Positive integer > 0	Namespace of the page
page_title	String, max. 255 characters	Title of the page
page_restrictions	Binary string	Comma-separated set of permission keys indicating who can move or edit this page (last revision)

Table 3.2: Fields and values included in table *people* as defined in WikiDAT

Field name	Possible values	Description
rev_user	Positive integer > 0	Unique identifier of registered user
rev_user_text	String, max. 255 characters	Nickname of a registered user

Table 3.3: Fields and values included in table *revision* as defined in WikiDAT

Field name	Possible values	Description
rev_id	Positive integer > 0	Unique identifier of a revision
rev_page	Positive integer > 0	Unique identifier of page modified in this revision
rev_user	Integer: -1, 0 or positive value	Unique identifier of registered user who performed this revision. -1 values indicate that user identifier was not recorded in the dump file; 0 values indicate anonymous users; positive values indicate identifier of registered users
rev_timestamp	Date and time value (YYYY-MM-DD HH:MM:SS)	Timestamp for recording this revision in the database
rev_len	Positive integer > 0	Length in characters of the wiki page after this revision
rev_parent_id	Positive integer > 0 or NULL	Link to the numerical identifier of the previous revision undertaken in the same wiki page. NULL value indicates first revision for a wiki page
rev_is_redirect	Binary (1 or 0)	1 value indicates that the page modified in this revision is a redirect
rev_minor_edit	Binary (1 or 0)	1 value indicates that the user marked the <i>minor edit</i> tick for this revision
rev_fa	Binary (1 or 0)	1 value indicates that, after this revision, the modified page displays the <i>feature article</i> status tag (locale dependent).
rev_comment	String (max. 255 characters)	Comment inserted by the user who performed this revision

Table 3.4: Fields and values included in table *logging* as defined in WikiDAT

Field name	Possible values	Description
log_id	Positive integer > 0	Unique identifier of login actions
log_type	String	Type of log action performed

log_action	String	Concrete action undertaken (within a given type)
log_timestamp	Date and time value (YYYY-MM-DD HH:MM:SS)	Timestamp for recording this log action in the database
log_user	Positive integer > 0	Unique identifier of registered user
log_username	String	Nickname of user who carried out the log action
log_namespace	Positive integer > 0	Namespace of the page on which the log action was performed
log_title	String (max. 255 characters)	Title of the page receiving the log action
log_comment	String (max. 255 characters)	Comment of the log action
log_params	String (max. 255 characters)	Additional descriptive params providing metadata about the log action (e.g. duration of a block)
log_new_flag	Positive integer > 0	For Wikipedias with <i>flagged-revisions</i> enabled, rev_id of the last flagged version of this page
log_old_flag	Positive integer > 0	For Wikipedias with <i>flagged-revisions</i> enabled, rev_id of the previous flagged version of this page

The code of WikiDAT can be retrieved from the project site on Github[[TODO:link]]. The project files include some example implementations of common analyses conducted on Wikipedia data, for descriptive purposes. The documentation and source code in these examples will be improved progressively to provide didactic case studies for other researchers interested in following similar approaches in their own analyses. In the last chapter of this document we visit some of these examples to illustrate some of these example cases.

The following prerequisites must be satisfied to run the examples included in this document, using the code included in WikiDAT:

- MySQL server and client (v5.5 or later).
- Python programming language (v2.7 or later, but not the v3 branch) and MySQLdb (v1.2.3).
- R programming language and environment (v 2.15.0 or later).
- Additional R libraries with extra data and functionalities:
 - *RMySQL* ¹⁸.
 - *lattice* (by Deepayan Sarkar).
 - *car* (by John Fox et al.).

¹⁸<http://cran.r-project.org/web/packages/RMySQL/index.html>

- *DAAG* (John Maindonald and W. John Braun) .
- *Hmisc* (by Frank E Harrell Jr, with contributions from many other users).
- *rjson* (by Alex Couture-Beil).

Please, refer to Chapter 4 introducing open source tools for data analysis for additional information about obtaining and installing these dependencies.

Part II

Conducting Wikipedia data analysis

Chapter 4

Methodology for Wikipedia data analysis

The methodology for Wikipedia data analysis is not very different from the general process to conduct data analysis in other fields. We must first retrieve our data source files (or retrieve the information from other online sources/APIs). Once the data is stored locally, we should undertake some sanity checks to ensure that the data has been retrieved properly. Preliminary EDA (Exploratory Data Analysis), including graphics summarizing important data traits should follow. Extra caution must be taken to identify possible missing values or any other odd/extreme values that may alter the results of our analysis. Finally, we prepare the data to obtain any intermediate results needed for our study, undertake the analysis (model building and refinement) and interpret the conclusions. This process may be iterative in case that we discover additional insights that could help us improve the model, although we must also pay attention to avoid relying too much on a single data set to build our model, since we could lose generality.

In this chapter, I offer some tips and advice that can help you to avoid common pitfalls while retrieving and preparing your Wikipedia data.

4.1 The big picture

As I have commented previously, no less than 75% of the total time for data analysis is consumed in the data retrieval and data preparation part. In the case of Wikipedia data analysis, this may increase up to 80-85%, depending on the actual type of analysis that we want to conduct. For studies at the macroscopic level, the proportion of time for data preparation will be closer to the lower bound. However, for fine-grained or high-resolution analysis, involving huge data sets, and possibly multiple large data sets in longitudinal studies, the scenario can be very complex.

Incredibly, only few books cover the very important task of data cleaning and data preparation. A well-known reference for many years, *Data preparation for data mining*[[TODO:REF!]] is now out-of-print and it can be found only through second-hand resellers. A more recent book, *Best Practice in Data Cleaning*[[TODO:REF!]] has been recently published to close this gap, dealing with important topics such as missing values and transformations. Some other books also cover this important stage as part of complete methodologies for applying certain types of analysis. For example, I can recommend *An R companion to applied Regression*[[TODO:REF]] and *Regression Modeling strategies*[[TODO:REF]] for linear models, along with a new reference covering these aspects for survival and event history analysis[[TODO:REF]]. All the same,

many applied statistics books still work with synthetic or already prepared data sets to dive into the actual details of the stastical tools and techniques as fast as possible, overlooking this stage.

The following sections will help you to find your way through the preparation of Wikipedia data for your own analyses.

4.2 Retrieve and store your data locally

In the first part, we presented different sources from which we can retrieve our data for Wikipedia analysis. Tools like WikiDAT can save us time to build this local store of Wikipedia data.

At the time of writing, WikiDAT can retrieve information from both *pages-meta-history* and *pages-logging* dump files. Together with the *user-groups* dump, that can be directly imported in a MySQL database, these 3 data sources can be combined to undertake multiple and interesting analyses on any Wikipedia language. You can consult again Section 3.8 for additional details about the data fields currently retrieved or computed by WikiDAT.

All you need to do is to find the link for these files in the Wikimedia Download center (see Section 1.7). Then, open a terminal and go to the *parsers* folder in the WikiDAT code. You can execute the following commands to retrieve the data from the *pages-meta-history* dump file (this will be automated in new versions of WikiDAT):

```
jfelipe@blackstorm:~/WikiDAT/sources$ mysql -u root -ppassword
mysql> create database wkp_lang;
mysql> exit
jfelipe@blackstorm:~/WikiDAT/sources$ mysql -u root -ppassword
wkp_lang < tables-wikidat.sql
jfelipe@blackstorm:~/WikiDAT/sources$ python pages_meta_history.py
wkp_lang lang lang-YYYYMMDD-pages-meta-history.xml.7z log_file.log
```

This will create a new MySQL database, the tables to store the information, and finally executes the parser on the downloaded dump file. You need to you need to change *password* with the actual password of the root user in MySQL (you can also create new users, see the manual section for this task). Please, note that there is a blank space in between *-u* and the user name but no blank space in between *-p* and the user password.

In the last command, change *wkp_lang* for the name of your database, *lang* with the name of the lang repository (e.g. eswiki, frwiki, enwiki, dewiki, etc.). The last two arguments are the name of the dump file and the name to create a log file that will store any warning or error messages (in case that any of these are created along the process). You can read regular traces to follow the progress of the parsing process. Once it has finished, you can check that the numbers of pages and revisions retrieved concur with the information on the download page.

To retrieve information stored in the *pages-logging* file, execute:

```
jfelipe@blackstorm:~/WikiDAT/sources$ python pages_logging.py
wkp_lang lang-YYYYMMDD-pages-logging.xml.gz log_file.log
```

Finally, to import in your local database information about user groups and privileges, retrieve the file for the Wikipedia language of your interest and type:

```
jfelipe@blackstorm:~/WikiDAT/sources$ gzip -d
lang-YYYYMMDD-user_groups.sql.gz
```



```
jfelipe@blackstorm:~/WikiDAT/sources$ mysql -u root -ppassword
wkp_lang < lang-YYYYMMDD-user_groups.sql
```

For example, if we are interested in recovering data from the dump files created on 2012-06-01 for the French Wikipedia:

```
jfelipe@blackstorm:~/WikiDAT/sources$ mysql -u root -ppassword
mysql> create database frwiki_062011;
mysql> exit
```

```
jfelipe@blackstorm:~/WikiDAT/sources$ mysql -u root -ppassword
frwiki_062011 < tables-wikidat.sql
jfelipe@blackstorm:~/WikiDAT/sources$ python pages_meta_history.py
frwiki_062011 frwiki frwiki-20120601-pages-meta-history.xml.7z log_file.log
```

```
jfelipe@blackstorm:~/WikiDAT/sources$ python pages_logging.py
frwiki_062011 frwiki-20120601-pages-logging.xml.gz log_file.log_file
```

```
jfelipe@blackstorm:~/WikiDAT/sources$ gzip -d
frwiki-20120601-user_groups.sql.gz
jfelipe@blackstorm:~/WikiDAT/sources$ mysql -u root -ppassword
frwiki_062011 < frwiki-20120601-user_groups.sql
```

Now, your data is ready for additional preparation and/or cleaning.

4.3 Routinary tasks for data cleaning and preparation

Below, you can find some useful tips to prepare Wikipedia data for your own analysis:

- Keep data preparation in the database: When we face the task of data preparation we usually have the option to prepare our data in the database, and then import them in our favourite analysis environment, or retrieve the raw data and undertake any data preparation and rearrangement tasks outside the database. In my experience, keeping all data preparation tasks in the database, as much as possible, is usually much faster and less error prone.
- *Anonymous editors*: The only identifier recorded in the database for anonymous editors is the IP address of the computer from which they performed the revision. No matter how many claims you find telling you otherwise, we cannot use this information to identify individual anonymous editors accurately. This involves some technical details about the inner features of Internet communication protocols, but suffice to say that many (really, many) computers can share the same IP address (from the point of view of the Wikipedia system receiving incoming requests). Outgoing connections can pass through proxies and other traffic filtering machines, thus showing up in the destination with the same IP address. As an example, consider the case in which Wikipedia inadvertently blocked an IP address that was performing vandal actions, banning *de facto* Wikipedia editing from the whole country of Qatar, in which all outgoing connections go through the same Internet Service Provider ¹.

¹https://en.wikinews.org/wiki/Qatari_proxy_IP_address_temporarily_blocked_on_Wikipedia

- *Bots and extremely active editors*: Sometimes, our study focuses on the activity of Wikipedia editors, and one of the common metrics found in research literature is the number of revisions per editor. In this case, an initial step should always be eliding all revisions performed by bots, special programs undertaking routinary operations, sometimes at a ver fast pace. For this, we can use the information in the *user_groups* table, looking for users in the group *bot*. However, we must also take some caution not to confound these bots with extremely active users, some of which can be incredibly prolific regarding the total number of revisions that they have contributed to Wikipedia ².
- *Missing editor information*: In certain dump files, we can find revisions for which the user identifier is missing. This can be caused by multiple reasons, not necessarily by errors in the dump process. For example, if a user account is completely deleted, its associated revisions may still be present in the database. In WikiDAT, I have marked these revisions with a -1 value in the *rev_user* field.
- *Widespread definitions*: The Wikipedia community usually follow some well-known definitions to classify editors, many of them derived from the assumptions used in the official stats page ³. For example, an *active wikipediaian* is a registered user who performed at least 5 revisions in a given month. Likewise, a *very active wikipediaian* is a registered user who made more than 25 revisions in a given month. Sometimes, when we wan to restrict our study to wikipediaians who have shown a minimum level of commitment with the project (in terms of activity) a usual approach is to take all registered users with at least 100 revisions along their whole lifetime in the project. This is the minimum threshold required to participate in several community voting processes.

4.4 Know your dataset

On the course of your own data analyses you will probably find situation in which problems or unexpected errors arise in unsuspected ways. This is a common, but again not frequently publicized situation for data analysts. In the end, the most precious weapon to overcome these hidden obstacles is learning as many details as possible about your data set and its generation process.

As an example, I am going to share with you some insights from a recent event history analysis on Wikipedia data. For this type of analysis, the researcher is usually interested in measuring time intervals between consecutive states of interest in different subjects. In the process of preparing the data tables to perform this analysis, I found useful details about the way in which MediaWiki software records revisions in the database.

Suppose that we want to measure time intervals between consecutive revisions performed on the same Wikipedia page. To calculate this info automatically, we can confront two tables: the first with all revisions for that page, ordered by their timestamp, except for the last one; the second with all revisions for that page, ordered in the same way, but this time removing the first one. In this way, we are facing each revision with the next one, for this page ⁴. This procedure will work whenever we can guarantee that every revision for the same wiki page has a different timestamp. Is this the case? The answer is yes, it is. MediaWiki software does not allows for concurrent editing of wiki pages (unlike GoogleDocs or similar technologies for

²http://en.wikipedia.org/wiki/Wikipedia:List_of_Wikipedians_by_number_of_edits

³<http://stats.wikimedia.org>

⁴We could also use the *rev_parent_id* to solve this problem

online collaborative editing. As a result, we can not expect to find two revisions for the same page with the same timestamp.

Now, we turn to the same study, but this time focusing on registered users, instead of wiki pages. For each user, we want to calculate the time interval between consecutive revisions. We can proceed in the same way, but this time our approach will fail. Why? Well, if we carefully inspect the entries, we will find that, for some users, we can find two, three or even more revisions sharing the same timestamp. How is that possible, anyway? The answer can be found in the values of the *rev_is_redirect* field. In this case, whenever a user creates a new redirect for a wiki page, two (or more, depending on how many redirects are created) entries in the database are generated, all sharing the same timestamp: one for a revision concerning the original page for which the new redirect is created, and one for every new redirect page that has been created. Therefore, we must impose additional filtering (for example, grouping by user and timestamp) to ensure that our data preparation process works.

In summary, you should always perform sanity checks of intermediate results (including descriptive graphs) to make sure that your scores make sense, and no hidden problems can jeopardize the results and conclusions from your analysis.

Chapter 5

Open source tools for data analysis

Nowadays, data analysts can choose among a wide variety of open source tools and programming languages to implement their studies. In fact, open source tools for data analysis and High-Performance computing are quickly becoming the preferred solution for many practitioners, scholars and professionals in this field.

In this chapter, I recap some essential open source tools that you can use to accomplish this endeavour. For sure, this list is far from complete, and I have only focused on the tools integrated in WikiDAT, that we will revisit later when we examine the practical case examples for Wikipedia data analysis. Furthermore, I have left aside any tools linked with distributed computing solutions such as Hadoop ¹ (based on the *map-reduce* programming paradigm) or some of its associated projects such as Pig, Cassandra, Hive or Mahout, to cite but a few instances. If you have read the previous chapters of this document, I strongly support the thesis that these tools, despite their increasing popularity, come with a high cost in the form of a steep learning curve, time and effort to effectively parallelize complex tasks. Actually, some associated Hadoop projects like Pig or Mahout try to alleviate this problem, providing additional abstraction layers to hide the inherent complexity of programming map-reduce processes.

Hence, we will not focus on this kind of tools at this time, although they may be eventually introduced and described in later versions of this document, to undertake high-resolution analyses on really huge data sets.

5.1 Python

Python is a general-purpose and multi-platform programming language that will act as a sort of “glue code” to assemble different pieces together and implement the software to fit our analysis.

One of the great advantages of Python is that it offers an extremely informative and complete online documentation for the baseline programming language, as well as presenting many common libraries available to automate common tasks. Browse the Python documentation ² to discover it by yourself!

5.1.1 Installing Python in GNU/Linux

If you are a Linux user, it is very probable that Python already comes installed in your favourite distribution, as it has become quite a hard requirement for many basic applications. To check

¹<http://hadoop.apache.org/>

²<http://www.python.org/doc/>

this, open a terminal and write:

```
jfelipe@blackstorm$ python
Python 2.7.2+ (default, Oct  4 2011, 20:06:09)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

If you can see this (the command line of the Python interpreter) then Python is installed in your system. To quit the interpreter, press `Ctrl+D`.

5.1.2 Installing Python in Windows or Mac OS

Since Python is a multi-platform programming language, you can also find an installer targeting different versions of other operating systems. Just point your browser to the Download Python page³ and find out the installer file that matches your flavour and version.

5.1.3 The *mysql-python* API

MySQLDB is a Python library that brings a convenient API to access MySQL from Python. The user's manual⁴ as well as the project page on SF.net⁵ offer additional information. You must consult the README file in the source package and follow the instructions to install it in your system. Binary files are also available for some GNU/Linux distributions. For instance, in Debian and Ubuntu you only need to install the package *python-mysqldb* to get it working on your system.

5.2 NumPy, SciPy and matplotlib

NumPy is the basic library in the Python programming language for mathematical operations and scientific computing⁶. In addition to this, *SciPy*⁷ delivers a comprehensive toolset for scientific programming in Python, covering many disciplines. Finally, *matplotlib*⁸ is a library to create high-quality graphics in a variety of formats, complementing NumPy and SciPy with graphical features. Jointly, these libraries conform a full-featured environment for scientific programming that can be extended even further with any of the available *scikits*.

5.3 Python Scikit-learn

The *scikits* are Python libraries providing even more functionalities to the baseline framework of NumPy and SciPy. Among these scikits, *scikit-learn* stands out as a very powerful library for data mining and machine learning⁹. The project is supported by Inria and Google, and currently implements a long list of different algorithms and tools for supervised and unsupervised learning, as well as data loading, data transformation and data visualization (including

³<http://www.python.org/getit/>

⁴<http://mysql-python.sourceforge.net/MySQLdb.html>

⁵<http://sourceforge.net/projects/mysql-python/>

⁶<http://numpy.scipy.org/>

⁷<http://www.scipy.org/>

⁸<http://matplotlib.sourceforge.net/>

⁹<http://scikit-learn.org/stable/>

3D support with RGL graphs). Future versions of WikiDAT will use this library to illustrate the application of machine learning techniques on Wikipedia data, along with implementations in the R programming language (that we are introducing very soon).

5.4 Database engine: MySQL

MySQL is a lightweight but powerful relational database engine software. The MySQL manual ¹⁰ explains in detail how to get and install ¹¹ MySQL in multiple operating systems and platforms. In Debian and Ubuntu GNU/Linux, you can just install the packages *mysql-client* and *mysql-server*. In the installation process, you will be prompted to introduce a password for the root user of MySQL. Please, take your time to select a password that you can remember later on.

Of course, MySQL is not the only open source database engine available for these purposes. After the acquisition of Sun Microsystems by Oracle Corporation, some companies has started to offer alternative storage engine and services based on MySQL, like Percona ¹². Yet another options is using PostgreSQL ¹³, another powerful open source database software with several different libraries to communicate with Python applications ¹⁴.

5.5 R programming language and environment

R a free software which offers the most complete and most powerful statistical programming language and environment available today. The R project website ¹⁵ is the entry point to the R world. A core development group of 20 people ¹⁶, who founded the R Foundation ¹⁷ to oversight the good progress of this project, are responsible for maintaining and improving the core environment.

Additionally, a very active ecosystem of developers and contributors are constantly augmenting the R suite of tools and features providing add-on packages known as R libraries. These libraries are published via the Comprehensive R Archive Network (CRAN) ¹⁸, a network of mirror servers that provide access to this repository from many different locations. At the time of writing these lines, CRAN lists more than 3,800 libraries and this number continues to grow exponentially.

5.5.1 Installing R

R is also a multi-platform programming language and statistical environment. You can read the R FAQ to find instruction about how to get and install R in your own computer.

- For GNU/Linux users, search your software management system to find out the R binaries. Make sure that you install the base environment (in Debian-like systems package *r-base*) and recommended packages (in Debian-like systems package *r-recommended*).

¹⁰<http://dev.mysql.com/doc/refman/5.5/en/index.html>

¹¹<http://dev.mysql.com/doc/refman/5.5/en/installing.html>

¹²<http://www.percona.com/software/>

¹³<http://www.postgresql.org/>

¹⁴<http://wiki.postgresql.org/wiki/Python>

¹⁵<http://www.r-project.org/>

¹⁶<http://www.r-project.org/contributors.html>

¹⁷<http://www.r-project.org/foundation/main.html>

¹⁸<http://cran.r-project.org/mirrors.html>

There are binary files¹⁹ available for all major distributions including RedHat, Suse, Debian and Ubuntu. The primary way to interact with R in GNU/Linux is to simply type (write `q()` and press Enter to exit):

```
jfelipe@blackstorm$ R
R version 2.15.0 (2012-03-30)
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors. Type 'contributors()'
for more information and 'citation()' on how to cite R or R packages in
publications.
Type 'demo()' for some demos, 'help()' for on-line help, or 'help.start()'
for an HTML browser interface to help.

Type 'q()' to quit R.
>
```

- For Windows users, please follow the links for Windows²⁰ or Windows64²¹ to install the base environment. This should already come with the recommended libraries plus a nice editor to avoid using the command-line interface (usually hidden from the eyes of standard users in Windows). You can open this GUI double-clicking on the new icon that should appear now on your desktop.
- In the case of Mac OS or Mac OSX users, please follow the installation instructions for Mac users²² in the R FAQ, that will point you to the adequate binaries to get R running on your fashion hardware. In this case, the binaries also install a simple graphical interface to interact with the R command interpreter.

5.5.2 Installing additional R libraries

The easiest way to install additional R packages is to do it inside R itself. Thus, execute either R from the GNU/Linux command-line or double-clicking the icon on your desktop in other operating systems.

On the R command interpreter, you can now type the following:

```
> install.packages("name_of_package", dep = T)
```

With this command, we request R to install the package whose name comes in double quotes as the first item in the brackets, and also to retrieve and install all necessary dependencies (other R packages) required for this package to work. If you are installing your first package in this session, you will be prompted to choose one of the many CRAN mirrors around the

¹⁹<http://www.vps.fmvz.usp.br/CRAN/bin/linux/>

²⁰<http://www.vps.fmvz.usp.br/CRAN/bin/windows/>

²¹<http://www.vps.fmvz.usp.br/CRAN/bin/windows64/>

²²http://www.vps.fmvz.usp.br/CRAN/doc/FAQ/R-FAQ.html#How-can-R-be-installed-_0028Macintosh_0029

world to obtain the additional software. Just select the closest mirror to your current location (or any other), then sit and relax while R does all the work for you. Following this procedure, can install any additional packages (a.k.a. R libraries) in your system.

In order to run the examples included in WikiDAT, you will need the following R libraries:

- *RMySQL* ²³.
- *car* (by John Fox et al.).
- *DAAG* (John Maindonald and W. John Braun) .
- *Hmisc* (by Frank E Harrell Jr, with contributions from many other users).
- *rjson* (by Alex Couture-Beil).

Regarding *RMySQL*, please check that MySQL server is already installed in your system. Specially for GNU/Linux systems, it is also recommendable to ensure that any additional dependencies are met installing this library directly from your favourite software management tool. For example, in Debian or Ubuntu you can install the package *r-cran-rmysql*.

5.5.3 Graphical user interfaces for R

A number of alternative GUIs have been created to facilitate the work with R, specially for novice users. The following are some suggestions that can help you to get a better R experience.

- *RStudio* ²⁴ (good for new R users, also allows working with a remote server).
- *Rcommander* ²⁵ (great for novel users, it automates many common analyses).
- *Rattle* ²⁶ (ideal for exploring R data mining capabilities).

5.5.4 R documentation and further references

One of the notable advantages of R is the abundancy of online, freely accessible documentation about the baseline environment and many of the CRAN libraries. In the last versions of R, there has been a remarkable concern among developers (encouraged by the R Core Team) to improve the companion documentaion for many of these packages.

The following options are essential for novel and experienced R users alike to learn how to use these libraries and solve their doubts:

- R Manuals: contributed documents and manuals from R users.
 - <http://cran.r-project.org/manuals.html>.
 - <http://cran.r-project.org/other-docs.html>.
- R Seek: Search in help files and R mailing lists.
 - <http://www.rseek.org/>.

²³<http://cran.r-project.org/web/packages/RMySQL/index.html>

²⁴<http://rstudio.org/>

²⁵<http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/>

²⁶<http://rattle.togaware.com/>

- R Task Views.
 - <http://cran.r-project.org/web/views/>
- The R Journal.
 - <http://journal.r-project.org/>
- Blogs and websites.
 - R-bloggers: <http://www.r-bloggers.com/blogs-list/>.
 - R Forge: <https://r-forge.r-project.org/>.
 - Quick-R: <http://www.statmethods.net/>.
 - R graphical manual: <http://rgm2.lab.nig.ac.jp/RGM2/images.php?show=all&pageID=363>

Finally, there is a very long list of published books around R, and a comprehensive review of these references falls beyond the scope of this document. All the same, there exist some well-known references adequate for R newcomers, as well as for R users with some experience willing to explore additional features and tools.

[[TODO: A bunch of references should be inserted below!!!]]

At the introductory level, one of my favourite references that I keep on recommending is the excellent book *Introductory Statistics with R*, by Peter Dalgaard. Pete is one of the members of the R Core Team, and the book is written in a concise and very pragmatic way. However, one minor disadvantage is that, despite he introduces some details of basic statistical techniques, this short manual is not intended to be a statistical reference for novel students or practitioners. Fortunately, this gap has been filled by a very recent book, *Discovering Statistics Using R*. The aim of this thick volume (~ 1000 pages) is to present essential statistical tools and methods for researchers in an accessible (and quite irreverent) style. The book is specially suitable for researchers and practitioners in social and experimental sciences, though it also covers many aspects pertaining observational (or correlational) studies.

Once you have acquired some basic skills about statistics and R programming, some references may help you to expand your knowledge and explore the many possibilities that R can offer us. *R in a nutshell* is a valuable companion manual that serves both as an catalogue of available tools in R and reference to find hands-on examples to use these tools. It also includes many pointers to data mining features and R libraries dealing with these techniques. A classic (but up-to-date) reference to learn effective methods for data analysis with R is *Data Analysis and Graphics Using R*, now in its 3rd edition. *Linear Models with R*, by J. Faraway remains as the authoritative introduction to linear models in R. There is also a prior, shorter and free version of this book available on CRAN manuals. Finally, *A Handbook of Statistical Analyses Using R* packs a concise reference for the implementation of many common statistical analyses and techniques for those users with solid theoretical background.

Chapter 6

Example cases in Wikipedia data analysis

6.1 Activity metrics

The first example case that we are going to review is the production of overall activity trends in Wikipedia. Some of these results are similar to those produced in <http://stats.wikimedia.org>, as well as in some well known research papers, such as [\[\[TODO:ref Voss paper\]\]](#) and [\[\[TODO:ref Almeida paper\]\]](#). This work is also based on the overall trend statistics included in my PhD. dissertation [\[\[TODO:ref Thesis Felipe\]\]](#).

6.1.1 Questions and goals

The evolution of the activity registered in Wikipedia has been one of the first topics of interest for quantitative researchers. As usual, the main spot of attention has been the English Wikipedia, in particular for creating models that try to explain and predict the activity trends in this language for the mid-term (see section 6.1.4 for more information).

The main goal has been to characterize the evolution of editing activity in Wikipedia that nurtures its fantastic growth rate (in content and number of articles). However, over the past 3 years the focus has shifted to explain the steady-state phase in which it has entered the activity in many larger Wikipedias, as well as possible reasons that may have influenced this change in the overall trends.

In the example, we will

6.1.2 Required data and tools

In the table *revision* generated by WikiDAT for any language, we can find the required data to undertake the analysis of overall activity trends. However, it is a sensible approach to use the information in the *user_groups* so that we can make a more accurate analysis by filtering out activity from *bots* (see section 4.3 for more information).

More precisely, we will use the following fields of the *revision* table:

- *rev_id* to count the number of revisions in a given time slot.
- *rev_page* to aggregate scores about number of pages.
- *rev_user* to aggregate scores about users, as well as to elid information about anonymous users and bots.

- *rev_timestamp* to track scores over time.
- *rev_fa* marking revisions with a FA label in them.

Additionally, we can also use information in the *page* table to break down scores by the namespace of pages, or to calculate the number of redirects (for articles). In this way, we can obtain for example how many revisions were received by encyclopedic articles (*main*), discussion pages (*talk*) and so forth.

6.1.3 Conducting the analysis

The directory *tools/activity*, in WikiDAT has code to generate numerical results and graphics summarizing general activity metrics at the macroscopic level for any Wikipedia language.

In the first place, it is necessary to retrieve the information for the 39 different Wikipedia languages included in the file *data_august_2011.py*, using the parsers as we have explained in Section 4.2. Then, we must prepare our data file in CSV format, running this Python script.

```
jfelipe@blackstorm:~/WikiDAT/sources$ python data_august_2011.py
```

This will generate the data file *data_082011.csv*. However, if you want to directly go ahead and run the R script file, you can use a version of this file that has been already computed and included in WikiDAT. In this case, the script computes a series of general descriptive metrics for each Wikipedia language as for August 2011. We can change the timestamp limits in the queries to produce results for other months as well (this process will be automated in forthcoming updates of WikiDAT. Finally you can load the R script *activity.R* in R studio and click on *Source* to produce the graphics and numerical results to describe our data. The results and graphics can be found in the *results* and *figs* folders, respectively.

6.1.4 Further reading

You can check similar approaches in a number of previous research works. In particular, Voss in[[TODO:ref measuring Wikipedia]] published in 2005 one of the first and most inspiring papers on general metrics and trends in Wikipedia. Almeida et al. [[TODO:Ref]] also studied different metrics to explain the evolution of Wikipedia. Finally, Chi et al. [[TODO:Ref singularity]] presented a model to explain the plateau phase in which many activity metrics in the English Wikipedia have entered. In my PhD. dissertation [[TODO:Ref]] I undertake a broader analysis spanning the top 10 Wikipedias according to the official article count by 2009.

6.2 The study of inequalities

6.2.1 Questions and goals

Contributions from registered users to Wikipedia are known to be highly unequal, with a small core of very active editors who perform a high proportion of all revisions[[TODO:ref]]. In the 10 largest Wikipedias by number of articles, this proportion may vary, but it is approximately around 10% of the total number of users who made 90% of all revisions in a given language.

At a smaller scale, the purpose of this example is to explore available tools in the *ineq* R package to study inequalities in the distribution of a certain statistic among the members of a population.

6.2.2 Required data and tools

In the WikiDAT folder *inequality* you will find two data files, *revisions.RData* and *users.RData* which you can use for experimentation. The R script *inequalities.R* make use of some tools like the Gini coefficient or the Lorenz curve to analyze the inequalities in the distributio of revisions made by wikipedians. In this case, as the examples data files has a small size, the inequality level is much less extreme that in the case of analyzing the whole population for one of the big Wikipedias. This is also due to the disproportionately high number of casual contributors that we find with respect to the short list of users included in the very active core of editors.

6.2.3 Conducting the analysis

In this case, you only need to load the file *inequalities.R* in RStudio and press *Source* to create the numerical results and graphs.

6.2.4 Further reading

In 2008 I published a paper[[TODO:Ref]] with Jesus G. Barahona and G. Robles a paper in the HICSS conference analyzing the evolution of the Gini coefficient for the top 10 Wikipedias by number of articles at that time. In that study, instead of evaluating the aggregate inequality level, we studied the inequality per month, that is, over the total number of contributions per month (instead of over the total number of contributes in the whole history of each language). In my PhD. dissertation[[TODO:ref]] I further expand the conclusions of this study.

Text

6.3 The study of *logging* actions

6.3.1 Questions and goals

Whilst the *pages-meta-history* dump files are by far the most popular of all Wikipedia database dumps, as we have seen there exist many more alternative data sources that we can use for our studies. Quite an interesting target can be the dump file storing all logged actions recorded on the database for every Wikipedia, including actions such as blocking users, protection of pages, and other administrative and maintenance tasks.

6.3.2 Required data and tools

In this case, we parse the dump file for the Simple English Wikipedia (*simplewiki*) to demonstrate some of the utilities in R to represent longitudinal data (data points over time) and build very simple models to analyze trends in these data. No special tools are required for the analysis in R, since all methods included in the script come with the standard installation.

6.3.3 Conducting the analysis

In the first place, we need to parse the logging table for *simplewiki*, using the tools provided in WikiDAT. Alternatively, you can download a compressed SQL file ready to be loaded in MySQL from http://gsyc.es/~jfelipe/WPAC_2012/simplewiki_logging_062012.sql.gz. Then, load and execute the R file *logging.R* in the *logging* directory of WikiDAT.

Bibliography