*data*

MDPI

*Data Descriptor*

# SNiPhunter: A SNP-Based Search Engine

**Werner P. Veldsman and Alan Christoffels ***

South African National Bioinformatics Institute, University of the Western Cape, Bellville 7535, South Africa
* Correspondence: alan@sanbi.ac.za; Tel.: +27-21-959-3645

Academic Editor: Vanathi Gopalakrishnan
Received: 10 June 2016; Accepted: 25 September 2016; Published: 29 September 2016

**Abstract:** Procuring biomedical literature is a time-consuming process. The genomic sciences software solution described here indexes literature from Pubmed Central's open access initiative, and makes it available as a web application and through an application programming interface (API). The purpose of this tertiary data artifact—called SNiPhunter—is to assist researchers in finding articles relevant to a reference single nucleotide polymorphism (SNP) identifier of interest. A novel feature of this NoSQL (not only structured query language) database search engine is that it returns results to the user ordered according to the amount of times a refSNP has appeared in an article, thereby allowing the user to make a quantitative estimate as to the relevance of an article. Queries can also be launched using author-defined keywords. Additional features include a variant call format (VCF) file parser and a multiple query file upload service. Software implementation in this project relied on Python and the NodeJS interpreter, as well as third party libraries retrieved from Github.

**Data Set:** SNiPhunter is available for online use at http://sniphunter.sanbi.ac.za, and can be downloaded from https://github.com/Werner0/SNiPhunter.



**Keywords:** search engine; reference SNP identifier; PMC-OAI; NoSQL; VCF

## 1. Summary

Reference single nucleotide polymorphism (refSNP) identifiers are used to earmark SNPs in the human genome. RefSNPs identifiers can be useful to include as terms in query constructs submitted to search engines when sourcing biomedical literature. These identifiers can be sourced by mapping the content of variant call format (VCF) files to refSNP identifiers. However, VCF files are not presented in a user-friendly format. Efficient extraction of refSNPs from these files may require the use of parsing software [1] such as PyVCF (a Python library) or VCFtools (a Perl library). This places an unfair expectation on researchers to familiarize themselves with scripting languages, and can therefore hamper rapid retrieval of biomedical literature during literature reviews. In addition, the amount of times a refSNP appears in an article is proof that the SNP is well-described in an article and thus linked to its topic, but search engines do not provide an option to return results based on such a quantification. Semantically enriching search engine results with a metric to indicate relevance of a suggested article will support the current trend [2] in transforming unstructured data into structured data, which in turn promotes the goal of designing decision support systems [3] that will assist non-technical researchers and clinicians in making informed choices. The vision of the application described in this article is to make a contribution towards realizing this goal. A cornerstone of the software design approach was to use free and open source software (F/oss) in combination with open access scientific literature. In addition to this, the development of this search engine steers clear

of traditional database design software that rely on structured query language (SQL) and the PHP hypertext preprocessor by using a NoSQL (not only structured query language) database supported by JavaScript and the NodeJS interpreter.

Conventions used to represent elucidated genetic data differ from one resource to another. The International Cancer Genome Consortium (ICGC) [4] lists data categories for variation together with their respective number of entries, and contains biospecimen-generated information on 12,979 donors. Of these, the database has simple somatic mutation (SSM) category data for 8038 individuals, potentially making the ICGC the ideal place to search for literature on a given SNP using its identifier (e.g., rs1799950—a SNP in the BRCA1 breast cancer-related gene). However, a search using this refSNP returns no results from the ICGC. When searching for this term using a more generic database, such as dbSNP [5] (which forms part of NCBI and therefore the Pubmed and Pubmed Central resource repositories), relevant results are returned for a plethora of information, after which data in related NCBI databases can then be retrieved by selecting a database from a drop-down list. Selecting Pubmed from the drop-down list provides the user with a subsequent list of articles that relate to the SNP, with the option to navigate to related articles; however, no indication is given as to the relevance of the article to the identifier if more than one article is associated with the identifier. Instead, literature is sorted with a preference for more recently publicized articles. Searching for the SNP by its identifier in non-institutionalized databases such as SNPedia [6] returns results that point the user to literature relevant to the SNP, with an indication of whether this literature is provided under open access agreements, but the order in which the results are listed does not give the user an indication of how the ranking was determined. In addition, SNPedia does not indicate the date of publication for the literature that it refers the user to. Interestingly, SNPedia does link the identifier with keywords and provides quantification by mentioning other SNPs in context (e.g., "This SNP, a variant in the BRCA1 gene, is 1 of 25 SNPs reported to represent independently minor, but cumulatively significant, increased risk for breast cancer"). Specific literature ordered by quantified relevance is not supplied by SNPedia, and bulk querying using reference SNP identifiers is not available as part of its user interface.

## 2. Data Description

A search engine called SNiPhunter was designed to address these shortcomings. It has an index of 69,463 unique refSNPs and 8743 author defined keywords in 20,650 biomedical articles from the Pubmed Central open access initiative. For comparison, the literature recommendation engine SNPedia [6], currently makes reference to 82,911 SNPs. The underlying NoSQL database allows RESTful access, and serves results via a web application and an application programming interface (API). API usage examples can be found at the SNiPhunter [7] website. The following ontological illustration (Figure 1) depicts the articles from Pubmed Central (PMC) open access initiative that were used to create SNiPhunter's data set. The ontology defines a SNiPhunter article in terms of its Pubmed membership. According to Pubmed [8], their database contains approximately 25 million indexed citations to life sciences journals and online books. Articles within Pubmed Central (a subdivision of Pubmed), contain about 3.7 million full free-text articles. Yet another subset of the PMC subdivision is the PMC-open access initiative (OAI) subset, which is made available to the public for data mining-related exercises. PMC corpus extraction results show the volume of this subset to be about 1.1 million. SNiPhunter's corpus is a third subset that contains only articles mentioning at least one refSNP (if the refSNP has a character count of between six and twelve). Currently, this refSNP subset contains about twenty thousand articles. Unfortunately, continuous automated bulk downloading of PMC-OAI content is prohibited, so any database derived from its content has to be manually updated on an intermittent basis. There are currently no scheduled updates with regards to the source content.

Information returned to the user if a refSNP is found within the SNiPhunter database include the following: refSNP, article title, publishing date, doi, Pubmed identifier, correspondence email address, and author defined keywords.

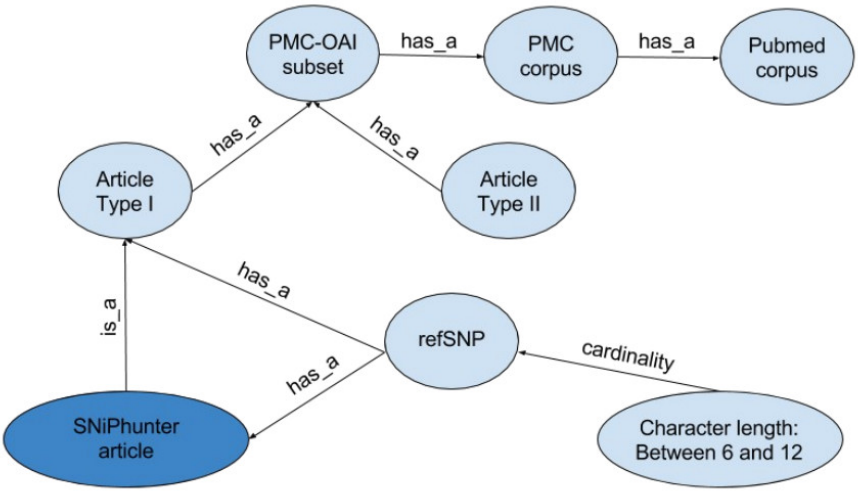


**Figure 1.** SNiPhunter's 69,463 unique reference single nucleotide polymorphism (SNP) identifiers and associated data were extracted from a corpus made available through the Pubmed Central open access initiative (PMC-OAI) subset of articles indexed with Pubmed.

## 3. Use Case

A random refSNP (rs16891982) was used as a query string in a use case test. The top result returned by SNiPhunter was compared to the top result returned by Pubmed and Google Scholar. Each of these three search engines returned a different most highly recommended article. While no implication as to the validity of the results is intended, Pubmed and Google Scholar did not give an explicit indication of the criteria used to determine the most relevant result nor how returned results were ordered. In contrast, it was clear from SNiPhunter's results that the result set was ordered according to the amount of times a refSNP occurred within an article. The top article recommended by SNiPhunter had thirteen occurrences of the refSNP identifier. Manual checking of the article validated SNiPhunter's recommendation, in that the article was highly focused on the refSNP identifier. The random term melanoma was then used as a query string to compare results from SNiPhunter's keyword search feature with results from the two other search engines mentioned above. Once again, the results were different for each of the sets returned, but SNiPhunter's results specifically recommended articles that contained at least one refSNP, whereas the other two search engines did not limit results to articles containing refSNP identifiers. SNiPhunter has found use in real world scenarios as well. A group of fourth year Medical Sciences students used the SNiPhunter refSNP search feature to source articles during a Human Genetics exercise. This was the first proof of SNiPhunter being used in practice. During a post-implementation survey, the web application's design appeal (the look and feel of the website) received an average score of eight out of ten. Of those surveyed, 85.7% reported that results returned using the refSNP search feature was useful. No user indicated that a feature of the website was too complex, although most users did not complete testing of all of the features. Real time results can be viewed online [9]. Visit SNiPhunter [7] to repeat these exercises and to explore other SNiPhunter features, such as the VCF file parser and multiple query upload service.

## 4. Methods

### *4.1. Parsing Semi-Structured Biomedical Literature*

Biomedical articles in XML format are made available in four batches via Pubmed Central's open access initiative (PMC-OAI) [10]. These articles were downloaded and filtered with the criteria that a given article contains a term starting with *rs*, or has a term with the second and third character being *rs* (to account for opening brackets). Extraction was carried out using string parsing instead of regular expressions, because a test of both methods on a 20,650 article subset of the literature revealed that string parsing roughly halved search time. Terms of interest (see data description) were extracted from the remaining PMC-OAI articles. Unicode control characters, commas, and backslashes were removed before binding terms to variables (to prevent fragmentation of article titles). RefSNPs containing less than six characters were excluded to avoid ambiguity arising from unrelated naming conventions (Figure 2). The number of occurrences of a refSNP in an article was counted by grouping together identical refSNP identifiers occurring within a given article. The extracted data was then converted to javascript object notation (JSON) and saved in text file format.

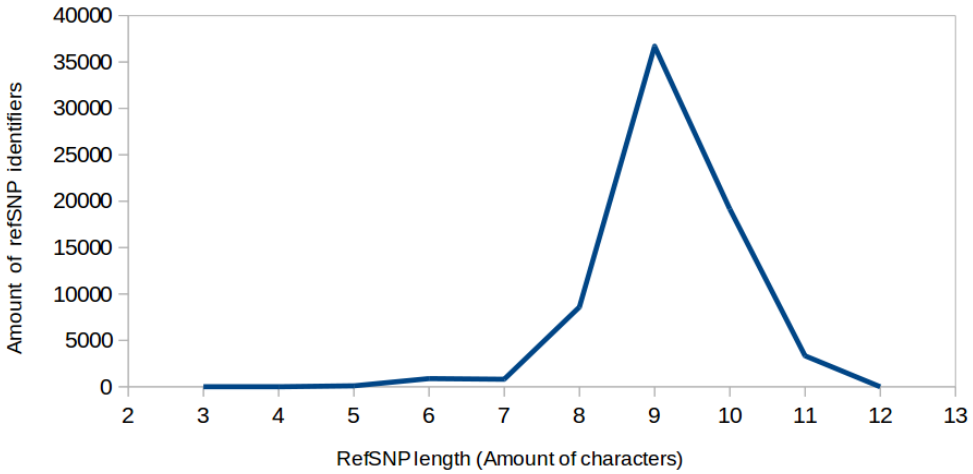


**Figure 2.** RefSNP (reference SNP) identifier character length follows a near-normal distribution between six and twelve characters. Potential refSNPs could therefore be constrained to this range during parsing of the corpus in order to avoid inclusion of pseudo refSNP identifiers.

There seemed to be a near-normal distribution of refSNP identifier character length (Figure 2). The trend in character length was used to negate ambiguity in certain *rs* terms observed while manually checking 109 of the lower character length *rs* terms. The curve was assumed to constitute a single naming convention (and therefore high precision), since it contained no additional peaks. Although this led to a loss of valid terms below a character length of six, and a lowering of recall, the loss was minimal compared to the total volume of extracted *rs* terms, as suggested by the low tail to the left of the near-normal distribution. The following diagram (Figure 3) illustrates the difficulties encountered when using automated testing to validate extracted terms. The outcome highlighted in red (left) indicates a case where a *rs* term in fact does not refer to a refSNP, but a similar refSNP ID exists in dbSNP. The other red outcome (right) indicates a case where the *rs* term does in fact refer to a refSNP ID, and the ID exists in dbSNP, but there are other naming conventions that use similar formats. The important outcome in this application is the first mentioned case. It is impossible to check every article manually, so the distribution was used with the hypothesis that the longer the character length of a refSNP ID is, the less likely it is that there would be ambiguity. A check of the 109 lower-end refSNP IDs (less than six characters in length) using a script to query the existence of a given *rs* term in the dbSNP database illustrated the difficulty with using automated article testing. It seemed like all but one (*rs2*) were false negatives. The false negative rate therefore appeared to be 0.155% ($108 \div 69463 \times 100$).

However, these excluded refSNPs are in SNiPhunter's database; they were just excluded from the predictive text feature. So, the real false negative rate is 0%. As an example, rs334 would not appear in the predictive text box, but would return results if the query was launched regardless. By excluding potential false positives from the predictive text feature, a user would not be presented with a given false positive unless a query was carried out, regardless of the set of possible *rs* terms shown by the predictive text feature. The predictive text feature was therefore used to mitigate the lack of context on extracted *rs* terms.

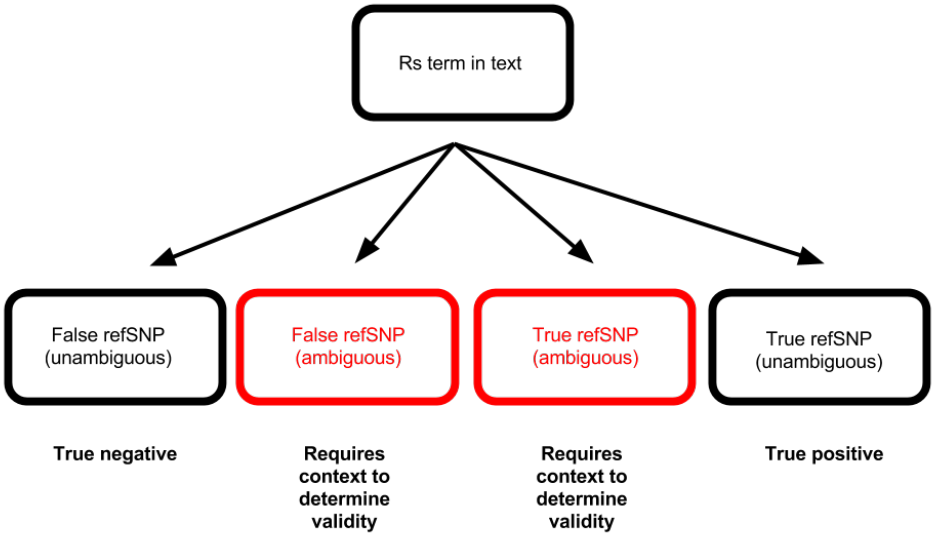


**Figure 3.** This diagram illustrates that overlapping naming conventions can lead to the inclusion or exclusion of invalid or valid extracted terms, respectively.

Sourcing articles from multiple databases could increase the scope of articles that can be retrieved, but the quality of free publications can vary widely. It should be noted that open access material is often placed into the following categories: gold (e.g., listed in the Directory of Open Access Journals [11]), green (e.g., listed in directories such as OpenDOAR [12] that caters to self-archived material), others (e.g., CiteSeerX [13]), and rogue material (papers published without consent). The separate issue of testing the validity of extracted keywords against alternatives to dbSNP presents problems of its own (Figure 3). Querying additional databases will have no effect on clearing ambiguities, because they arise from a lack of context for data extracted from the literature. Results from such exercises will never provide information on the context surrounding the extracted terms in the original source literature. Despite this lack of context, access to the full text of articles remains crucial. Figure 4 illustrates the value of full text access during data mining by saturating the occurrence frequency of gene names in the standard article subsections. It is clear that abstracts are the least informative. If cost is a limiting factor for a reader who is interested in determining the relevance of thousands of articles based on the occurrence of a gene of interest—as was the case in the study that the data for Figure 4 was drawn from—availability of free access to to all subsections of an article will be a primary determinant of the feasibility of a project.


**Figure 4.** Gene name occurrence frequency depicted with incremental saturation (after Shah et al. [14]). The frequency with which gene names occur in the standard subsections of scientific articles supports the argument for providing full-text access free of charge, specifically to benefit data mining-related exercises.

### *4.2. Implementing a Web Application and Web Service*

A website with separate pages for each of the application's features (single refSNP search, multiple refSNP search, keyword search, API guideline, and VCF file parser) was created with HTML, CSS (with Bootstrap), and javascript (with jQuery). Named entity extraction packages (such as MutationFinder [15] and tmVar [16]) are useful; however, the custom Python scripts used to extract data for inclusion in the SNiPhunter database allowed manual control over the literature extraction process. For example, the correspondence author email addresses, which is one of the defining features of SNiPhunter, could be extracted together with the other desired features mentioned above. There were also some peculiarities in the XML files that necessitated the implementation of, for example, a helper function (written in Python) to prevent fragmentation of article titles.

The Node interpreter, and third party packages (JSON-server [17], Awesomplete [18], Awk [19], Forever [20], Forever-service [21], Highlight [22], Intercept-stdout [23] and Multer [24]) downloadable from Github were installed to supplement the web application design. The JSON-server package provides a database server and API. SNiPhunter's NoSQL database is managed by a package called Lowdb [25] wrapped within the JSON-server package. However, a manually created flat-file, generated during literature extraction, contains the SNiPhunter database fields and entry values. Port forwarding was used to redirect traffic on port 80 to the JSON-server running on its internally-specified port 3000. In this way, the database was exposed as a web application and a web service.

### *4.3. Implementing a Database Interaction Protocol*

A representational state transfer (REST) protocol was used for interaction between the web application and the database, while calls to the database are initiated with asynchronous javascript and XML (AJAX). Returned data objects are sorted according to the amount of times a refSNP has occurred in an article, and the web application's document object model (DOM) is dynamically updated with search results. Data objects returned via calls to the API are not sorted, but the returned data can be further manipulated with the use of scripting languages such as Python. The API complies with RESTful standards, and provides access to the database using simple URLs (web addresses) concatenated to parameters, which allow the user to select relevant information from the database. Examples of how to use the API are available at the SNiPhunter website [7].

## 5. Conclusions

This application relies exclusively on PubMed's open access initiative. The intention is to illustrate that open access initiatives are making new types of applications possible. In that sense, it is a proof-of-concept search engine. Novel value is added by offering (i) results with an explicit indication of the amount of times a refSNP has been mentioned; and (ii) returning the author's correspondence address in the results. Author-defined keywords, appearing within the text of articles indexed by SNiPhunter, can also be used to query the database. The normal distribution of refSNP character

length might be of use to future developers dealing with ambiguity in naming conventions. We also found that most researchers confuse PMC-OAI with Pubmed and/or Pubmed Central. The ontological diagram in this manuscript focuses the readers attention on Pubmed's structure while at the same time advancing the concept of open access literature.

**Acknowledgments:** This work was supported by the South African Research Chairs Initiative of the Department of Science and Technology and National Research Foundation of South Africa.

**Author Contributions:** W.P.V. and A.C. conceived and designed the experiments; W.P.V. performed the experiments; W.P.V. and A.C. analyzed the data; W.P.V. wrote the first version of the paper; A.C. supervised the project and provided the funding for the project.

**Conflicts of Interest:** The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| API | Application programming interface |
| SNP | Single nucleotide polymorphism |
| NoSQL | Not only structured query language |
| SQL | Structured query language |
| VCF | Variant call format |
| refSNP | Reference single nucleotide polymorphism |
| PMC-OAI | Pubmed Central open access initiative |
| F/oss | Free and/or open source software |
| REST | Representational state transfer |
| XML | Extensible markup language |
| JSON | JavaScript object notation |
| HTML | Hypertext markup language |
| CSS | Cascading style sheets |
| AJAX | Asynchronous JavaScript and XML |
| DOM | Document object model |

## References

1. Danecek, P.; Auton, A.; Abecasis, G.; Albers, C.A.; Banks, E.; DePristo, M.A.; Handsaker, R.E.; Lunter, G.; Marth, G.T.; Sherry, S.T.; et al. The variant call format and VCFtools. *Bioinformatics* **2011**, *27*, 2156–2158.
2. Machado, C.M.; Rebholz-Schuhmann, D.; Freitas, A.T.; Couto, F.M. The semantic web in translational medicine: Current applications and future directions. *Brief. Bioinform.* **2015**, *16*, 89–103.
3. Price, W.N. Describing Black-Box Medicine. *B.U. J. Sci. Tech. L.* **2015**, *21*, 347.
4. Zhang, J.; Baran, J.; Cros, A.; Guberman, J.M.; Haider, S.; Hsu, J.; Liang, Y.; Rivkin, E.; Wang, J.; Whitty, B.; et al. International Cancer Genome Consortium Data Portal—A one-stop shop for cancer genomics data. *Database* **2011**, doi:0.1093/database/bar026.
5. Sherry, S.T.; Ward, M.H.; Kholodov, M.; Baker, J.; Phan, L.; Smigielski, E.M.; Sirotkin, K. dbSNP: The NCBI database of genetic variation. *Nucleic Acids Res.* **2001**, *29*, 308–311.
6. Cariaso, M.; Lennon, G. SNPedia: A wiki supporting personal genome annotation, interpretation and analysis. *Nucleic Acids Res.* **2012**, *40*, D1308–D1312.
7. SNiPhunter. 2016. Available online: http://sniphunter.sanbi.ac.za (accessed on 27 September 2016).
8. Pubmed. 2016. Available online: http://www.ncbi.nlm.nih.gov/pubmed (accessed on 27 September 2016).
9. Google form. 2016. Available online: https://goo.gl/ccGgwb (accessed on 27 September 2016).
10. Pubmed Central open access initiative. 2016. Available online: https://www.ncbi.nlm.nih.gov/pmc/tools/ftp/ (accessed on 27 September 2016).
11. Directory of Open Access Journals. 2016. Available online: http://doaj.org (accessed on 27 September 2016).
12. OpenDOAR. 2016. Available online: http://www.opendoar.org (accessed on 27 September 2016).
13. CiteSeerX. 2016. Available online: http://citeseerx.ist.psu.edu/index (accessed on 27 September 2016).
14. Shah, P.K.; Perez-Iratxeta, C.; Bork, P.; Andrade, M.A. Information extraction from full text scientific articles: Where are the keywords? *BMC Bioinform.* **2003**, *4*, 20.

15. Caporaso, J.G.; Baumgartner, W.A.; Randolph, D.A.; Cohen, K.B.; Hunter, L. MutationFinder: A high-performance system for extracting point mutation mentions from text. *Bioinformatics* **2007**, *23*, 1862–1865.
16. Wei, C.H.; Harris, B.R.; Kao, H.Y.; Lu, Z. tmVar: A text mining approach for extracting sequence variants in biomedical literature. *Bioinformatics* **2013**, *29*, 1433–1439.
17. Typicode. JSON-Server. 2015. Available online: https://github.com/typicode/json-server (accessed on 30 September 2015).
18. Verou, L. Awesomplete. 2015. Available online: https://github.com/LeaVerou/awesomplete (accessed on 30 September 2015).
19. Yu, G. Awk. 2015. Available online: https://github.com/guo-yu/awk (accessed on 14 December 2015).
20. Robins, C. Forever. 2015. Available online: https://github.com/foreverjs/forever (accessed on 30 September 2015).
21. Zapty Inc.; Agarwal, A. Forever-Service. 2015. Available online: https://github.com/zapty/forever-service (accessed on 30 September 2015).
22. Sagalaev, I. Highlight. 2015. Available online: https://github.com/isagalaev/highlight.js (accessed on 30 November 2015).
23. Farthin, S. Intercept-Stdout. 2015. Available online: https://github.com/sfarthin/intercept-stdout (accessed on 14 December 2015).
24. Yaapa, H. Multer. 2015. Available online: https://github.com/expressjs/multer (accessed on 14 December 2015).
25. Typicode. Lowdb. 2015. Available online: https://github.com/typicode/lowdb (accessed on 30 September 2015).