Bioinformatics, 32(11), 2016, 1755–1757
doi: 10.1093/bioinformatics/btw057
Advance Access Publication Date: 1 February 2016
Applications Note

Databases and ontologies

The geospatial data quality REST API for primary biodiversity data

Javier Otegui* and Robert P. Guralnick*

Florida Museum of Natural History, University of Florida, Gainesville, FL, USA

*To whom correspondence should be addressed.
Associate Editor: Jonathan Wren

Received on 26 November 2015; revised on 21 January 2016; accepted on 22 January 2016

Abstract

Summary: We present a REST web service to assess the geospatial quality of primary biodiversity data. It enables access to basic and advanced functions to detect completeness and consistency issues as well as general errors in the provided record or set of records. The API uses JSON for data interchange and efficient parallelization techniques for fast assessments of large datasets.
Availability and implementation: The Geospatial Data Quality API is part of the VertNet set of APIs. It can be accessed at http://api-geospatial.vertnet-portal.appspot.com/geospatial and is already implemented in the VertNet data portal for quality reporting. Source code is freely available under GPL license from http://www.github.com/vertnet/api-geospatial.
Contact: javier.otegui@gmail.com or rguralnick@flmnh.ufl.edu
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Primary Biodiversity Data (PBD) are the most basic and interpretation-free pieces of information upon which most biodiversity studies are built (Guralnick and Hill, 2009). They are characterized, at minimum, by a geospatial locality and a taxonomic identification (Johnson, 2007). Macro-scale biodiversity studies require larger PBD assembly, which are usually extracted from many different sources. To allow interoperability among these sources, data publishers are encouraged to store and share their records utilizing standard vocabularies such as the Darwin Core (Wieczorek *et al.*, 2012) and tools (Robertson *et al.*, 2014) for providing seamless access via portals such as VertNet (http://www.vertnet.org) and the Global Biodiversity Information Facility (http://www.gbif.org).

While quantity may be a requirement for scaling up analyses, quality assurance is equally important, as has been amply demonstrated in a set of recent studies. (Belbin *et al.*, 2013; Boakes *et al.*, 2010; Gaiji *et al.*, 2013; Maldonado *et al.*, 2015; Yesson *et al.*, 2007). While the scope of the problem has been made clear, solutions in the form of standard tools for data quality assessment have lagged behind, even if 'best practices' and 'principles' white papers are available (Chapman, 2005a, 2005b; Hill *et al.*, 2010).

Here, we present a set of functions that perform basic and advanced spatial quality assessments over user-provided sets of PBD, wrapped in a REST API. REST relies on HTTP requests to specific URL endpoints, to send and receive data to and from the underlying services, generally using JSON as the data interchange format. Our implementation focuses on best practice outputs and works at scale, leveraging cloud-based processing. It can be implemented at any point in the data cleaning workflow, from ingest by publishers and aggregators to end-users looking to further check downloaded datasets. Although the focus is on PBD, this service is usable for data coming from related domains such as ecological genomics, or wherever biological records have geospatial data.

2 Implementation

2.1 Input methods

The API currently accepts two request methods and is located at: http://api-geospatial.vertnet-portal.appspot.com/geospatial.

GET requests are best for assessing records one at a time. Each record is processed via its own HTTP request with parameters passed in the query string. Currently, the API works with the following four elements, all optional but which must be provided as shown if present: `decimalLatitude`, `decimalLongitude`, `countryCode` and `scientificName`.

POST requests are best for assessing larger amounts of records with a single call. With this method, the user sends a JSON array with one or more records (each one as a JSON object) in the body of the HTTP request. Records can have an arbitrary number of fields in each JSON object, but the API will only work on the four fields described above.

Examples on how to build both requests, and fully worked examples of inputs and outputs, are available in the Supplementary Material.

2.2 Data sources

The detection of many of the geospatial issues requires use of external spatial data sources. One of these data sets is the country administrative boundaries, extracted from the Global Administrative Areas database (http://www.gadm.org). Adding this source enables assessments comparing features between the provided coordinates and country.

The other data set is the collection of expert range maps compiled by the International Union for the Conservation of Nature (http://www.iucnredlist.org). It consists on a series of ESRI shapefiles which describe the known range of each species. Including this source enables spatio-taxonomic checks such as whether the point is located inside the range boundary, and if not, the distance between occurrence points and closest range map edge.

2.3 Process and output

The API makes three types of assessments on the supplied record. Completeness is an evaluation to what extent the provided information has enough detail (e.g. coordinates are present and have good precision, to at least two decimal places). Consistency is a check whether record information sources match (e.g. coordinates fall within boundaries of specified country or range maps, where available). Correctness is an assessment of common errors (e.g. coordinate swap). For a full list, see source code API documentation (http://bit.ly/1NkNDde).

Depending on the level of detail of the supplied records, the API will perform between three and fourteen different checks. Each one returns a true/false value indicating success or failure. At the end of the process, the results of all checks are formatted as a new JSON document, called `flags`, which gets inserted as a new field in the provided record. See the Supplementary material for an example of a `flags` object.

Several implementations improve performance: (i) data sources are properly indexed, (ii) all requests are cached and (iii) POST requests undergo significant parallelization by sending asynchronous requests for individual records and combining all results into a single response.

The final result of an API call depends on the request method. For GET requests, a JSON object with the four previously mentioned fields plus the new `flags` element is returned. If any of these four is missing, it will appear in the result with an empty string, and every other field will be ignored in the output. For POST requests, the API returns the input array, with the same fields and values, plus the `flags` element.

2.4 Limitations

The main concern against API performance is response speed for larger datasets. Some geometric calculations (like distance to range map) are computationally expensive and can be slow. Besides, the API is built on top of a Google App Engine (GAE) application, and instance initialization can delay the execution of the functions. Still, benchmark tests show an average of ~86 seconds to process a POST request of 1000 records. Also, due to third-party services limitations, there is a hard limit of 1000 records per POST request, to avoid unresponsiveness.

3 Discussion

Using an external API for data quality management, such as the one we present here, enables access to relatively complex assessment functions to data managers with little or no programming skills. The performed assessments codify best practices, grouped into completeness, consistency and correctness checks. The API integrates well with most programming languages and frameworks, so application developers can leverage these tools easily. A special benefit of using JSON as the data format is that information can stream seamlessly between systems and workflows.

For POST requests, we decided to include all the request fields in the response, instead of ignoring them as with GET requests. This gives more flexibility to the users, since they don't have to heavily modify the structure of their record set: they only need to make sure the four key fields are called according to the Darwin Core standard. Besides, this removes the hassle of including the quality flags on the original records *a posteriori*. Just as a simple example, one could download a set of records from the GBIF API (http://www.gbif.org/developer/occurrence) and, without modification, pass the result to the Geospatial Quality API to get the same record structure plus the quality flags.

GET requests work best when integrated into single-record viewers. One currently working example is the 'Spatial Quality' tab on VertNet's occurrence pages, where each record is checked against this API on the fly and results are formatted according to the web portal style (Supplemental Fig. S1). This provides an easy-to-read report per record for data consumers (http://vertnet.org/resources/spatialqualitytabguide.html). Our future plans include adding more assessments to the set of functions and developing further performance optimizations.

Acknowledgements

We thank Walter Jetz and Jeremy Malczyk, from Map Of Life, for their support and help with the management of the large database and spatial functions.

Funding

This research has been carried out as part of the Map of Life and VertNet projects, funded by the National Science Foundation (NSF) Grant nos. DBI-0960549, DBI-1062148 and DBI-1535793.

Conflict of Interest: none declared.

References

Belbin,L. *et al*. (2013) A specialist's audit of aggregated occurrence records: an 'aggregator's' perspective. *Zookeys*, **305**, 67–76.

Boakes,E.H. *et al*. (2010) Distorted views of biodiversity: spatial and temporal bias in species occurrence data. *PLoS Biol*., **8**, e10000385.

Chapman,A.D. (2005a) Principles and Methods of Data Cleaning – Primary Species and Species-Occurrence Data, version 1.0. Report for the Global Biodiversity Information Facility, Copenhagen. Available online at http://www.gbif.org/orc/?doc_id=1262.

Chapman,A.D. (2005b) Principles of Data Quality, version 1.0. Report for the Global Biodiversity Information Facility, Copenhagen. Available online at http://www.gbif.org/orc/?doc_id=1229.

Gaiji,S. *et al*. (2013) Content assessment of the primary biodiversity data published through GBIF network: status, challenges and potentials. *Biodivers. Inf*., **8**, 94–172.

Guralnick,R. and Hill,A. (2009) Biodiversity informatics: automated approaches for documenting global biodiversity patterns and processes. *Bioinformatics*, **25**, 421–428.

Hill,A.W. *et al*. (2010) *GBIF Position Paper on Future Directions and Recommendations for Enhancing Fitness-for-Use across the GBIF Network, Version 1.0*. Global Biodiversity Information Facility, Copenhagen. Accessible online at http://www.gbif.org/resource/80623.

Johnson,N.F. (2007) Biodiversity informatics. *Annu. Rev. Entomol*., **52**, 421–438.

Maldonado,C. *et al*. (2015) Estimating species diversity and distribution in the era of Big Data: to what extent can we trust public databases? *Glob. Ecol. Biogeogr*., **24**, 973–984.

Robertson,T. *et al*. (2014) The GBIF integrated publishing toolkit: facilitating the efficient publishing of biodiversity data on the internet. *PLoS One*, **9**, e102623.

Wieczorek,J. *et al*. (2012) Darwin Core: an evolving community-developed biodiversity data standard. *PLoS One*, **7**, e29715.

Yesson,C. *et al*. (2007) How global is the global biodiversity information facility? *PLoS One*, **2**, e1124.