tidygeocoder: An R package for geocoding

Jesse Cambon¹, Diego Hernangómez¹, Christopher Belanger¹, and Daniel Possenriede¹

¹ Independent Researcher

Summary

Tidygeocoder is a package for the R programming language (R Core Team, 2021) that allows researchers and analysts to easily perform geocoding. Geocoding (also called “forward geocoding”) is the process of obtaining geographic coordinates (longitude and latitude) from an address or a place name, while reverse geocoding is the process of obtaining an address or place name from geographic coordinates.

Forward and reverse geocoding play an important role in geospatial data analysis across many disciplines and are commonly performed through the use of web-based geocoding services, which are accessible as APIs (Kounadi et al., 2013). Geocoding was historically available only through commercial geographic information system (GIS) software that can be expensive and cumbersome, making web-based services an attractive free or lower-cost alternative (Karimi & Karimi, 2017). A specific geocoding service may perform better or worse for particular geographic regions or purposes, so there can be value in switching between services for cross-validation (Kılıç & Gülgen, 2020).

To use a geocoding service you must first execute an API query; then you need to extract and format the data received from the service and incorporate it into your project. However, geocoding services vary widely in their API parameters, capabilities, and output data formats, which can make it difficult for users to leverage a new service or switch between them. Tidygeocoder addresses this challenge by providing users with a simple and consistent interface for a number of popular geocoding services, so that users can spend less time worrying about data manipulation and API parameters and more time developing their projects. Tidygeocoder is actively used and cited in academic research and publications (Baumer et al., 2021; Décaire & Sosyura, 2020; Durbin, 2020; Hegde et al., 2021; King et al., 2020; Raymond et al., 2021; Walming et al., 2021).

Statement of Need

Tidygeocoder was created to remove obstacles that can make geocoding time-consuming and challenging. In contrast to other R packages that offer geocoding capabilities as part of a larger feature set or are built around a single geocoding service (Cooley, 2020; Gombin & Chevalier, 2021; Kahle & Wickham, 2013; Possenriede et al., 2021; Prener & Fox, 2021; Rudis & Thompson, 2018; Tennekes, 2021; Unterfinger, 2021; Walker, 2020), tidygeocoder is focused on providing access to many geocoding services through a standard interface.

The first challenge in geocoding is to construct an API query to a geocoding service. However, the APIs of geocoding services differ greatly. For instance, Nominatim, a geocoding service from the OpenStreetMap project (OpenStreetMap contributors, 2017), has separate street, city, state, country, postal code, and county parameters that can be used to specify...
components of an address. Other services such as Google only use a single address parameter to construct queries.

Additionally, the API parameter names are not standardized between services. The single-line address parameter for Nominatim is "q" (for query) while for Google it is "address". Some services such as Mapbox and TomTom use a non-standard query string format, which requires a different approach for constructing queries. Also, the same service can require a different API query and return output data in a different format depending on whether one input is given ("single input geocoding") or multiple inputs are given ("batch geocoding").

For reverse geocoding, some services such as Nominatim use separate latitude and longitude parameters, whereas other services combine latitude and longitude into a single parameter. Services can also require other parameters such as an API key and the desired output data format (e.g. JSON or XML).

Another challenge is the extraction and formatting of the API output. Geocoding services differ widely in what kind of data they return and how the data is structured. Working with this data therefore requires a variety of data manipulation work from the user. Services often return nested JSON data, but there is no standard format for this data, so users must locate the relevant data they wish to extract in the JSON structure and format it as needed.

**Functionality**

The tidygeocoder package provides a mechanism to utilize geocoding services through a unified interface and receive output data in a tidy dataframe format (Wickham, 2014) that can be easily incorporated into projects. A universal set of input parameters is mapped to the specific API parameters for each service and the relevant parts of the output data are extracted and formatted. This reduces the amount of time and effort required to use geocoding services and enables users to seamlessly transition between services.

For forward geocoding, users can provide addresses and place names with either a single parameter or multiple address component parameters (i.e. city, state, country, etc.). For reverse geocoding, the latitude and longitude parameters are specified with two separate parameters. These inputs can be provided standalone (i.e. a single value or vector) or within a dataframe.

Tidygeocoder limits the rate of API querying automatically based on the usage policy restrictions of the selected geocoding service. Only unique inputs are sent to geocoding services even if duplicate data is provided to avoid redundant or needlessly large queries. Built-in dataframes are used to store important information on geocoding services such as parameter names, query rate limits, and the maximum allowed size of batch queries. This makes these values transparent to users and allows developers to easily update them as needed. Some package documentation is directly generated from these dataframes to reduce the need for manual updates.

Tidygeocoder makes use of the httr package (Wickham, 2020) to execute API queries, the jsonlite package (Ooms, 2014) to convert JSON data returned from geocoding services into dataframes, the dplyr package (Wickham et al., 2021) for data manipulation, and the tibble package (Müller & Wickham, 2021) to return a tidy dataframe format (Wickham, 2014; Wickham et al., 2019).

**References**


