Package 'rgeolocate'

December 20, 2021

Type Package

Title IP Address Geolocation

Version 1.4.2

Date 2021-12-18

Author Os Keyes [aut, cre], Drew Schmidt [aut], David Robinson [ctb], Chris Davis [ctb], Bob Rudis [ctb], Maxmind, Inc. [cph], Pascal Gloor [cph], IP2Location.com [cph]

Maintainer Os Keyes <ironholds@gmail.com>

Copyright MaxMind, Inc. for the underlying libmaxmminddb library, IP2Location.com for libip2location, and the package authors for all other content.

Description Connectors to online and offline sources for taking IP addresses and geolocating them to country, city, timezone and other geographic ranges. For individual connectors, see the package index.

License Apache License (== 2.0)

BugReports https://github.com/ironholds/rgeolocate/issues

LinkingTo Rcpp

Imports Rcpp, httr

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.1.2

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-12-20 08:40:02 UTC

R topics documented:

`db_ip` *Geolocate IP Addresses Through db-ip.com*

Description

`ip_api` consumes a vector of IP addresses and geolocates them via db-ip.com. See 'Details' for more information.

Usage

```
db_ip(ip_addresses, key)
```

Arguments

`ip_addresses`	a character vector of IP addresses.
`key`	a db-ip.com API key.

Details

db-ip.com provides IP geolocation, either for free (with a cap of 2,500 requests a day) or in various paid tiers.

To use the service, register there for an API key, and plug that key and the IP addresses into `db_ip`

Value

a list of lists, each containing the geolocated values for one IP address. The variables found depend on the level of access your API key has; see the DB-IP API documentation on their website for more information.

See Also

`ip_api` and `ip_info` for other online geolocation APIs.

Examples

```
## Not run:
db_ip(ip_addresses = "173.194.67.1", key = "ThisIsNotARealKey")

## End(Not run)
```

`ip2location` *Geolocate IP Addresses through IP2Location Databases*

Description

IP2Location provides proprietary databases for IP geolocation; this function acts as a binding to them, letting you efficiently geolocate a vector of IP addresses to retrieve various values (much like `maxmind`). Note that depending on the database type you have, certain fields may or may not be available.

Usage

```
ip2location(
  ips,
  file,
  fields = c("country_code", "country_name"),
  use_memory = TRUE
)
```

Arguments

`ips`	A character vector of IP addresses.
`file`	The path to an IP2Location binary database. One is included in the package (see the examples below); full datasets can be purchased, and sample ones downloaded, at the ip2location website.
`fields`	Which pieces of metadata to retrieve for each IP address. Options are: • country_code: the ISO code of the country. • country_name: the English-language name of the country. • region_name: the English-language name of the region. • city: the English-language name of the city. • isp: The Internet Service Provider • lat: The latitude. • long: The longitude. • domain: The domain name associated with the IP (if any). • zip_code: The Zip Code or Post Code or national equivalent. • timezone: The timezone, in the format +02:00/-03:00 from UTC. • netspeed: The internet connection class of the IP address. • international_code: The international dialing code. • area_code: The local dialing code. • station_code: The identifying code of the nearest weather station. • station_name: The name of the nearest weather station. • mcc: The Mobile Country Code, which identifies mobile stations. • mnc: The Mobile Network Code, which (with MCC) uniquely identifies the mobile carrier.

- mobile_brand: The commercial brand associated with the mobile carrier.
- elevation: The elevation of the location above sea level, in meters.
- usage_type: The type of organisation or purpose behind the IP; see the list here.

Note that these fields may or may not be available depending on your database type.

`use_memory` Whether to cache the binary in memory or not. Caching it drastically increases the speed of geolocation, but may be too much for very old machines. Set to TRUE by default.

Value

A data.frame containing the geolocation metadata about `ips`; missing values are represented by `NA`.

See Also

`maxmind`, which uses MaxMind proprietary databases to get similar information.

Examples

```
file <- system.file("extdata","ip2_sample.bin", package = "rgeolocate")
example_ip <- "2A04:0000:0000:0000:0000:0000:0000:0000"

ip2location(example_ip, file, c("country_code", "country_name", "region", "city"))
```

---

ip_api *Geolocate IP Addresses Through ip-api.com*

Description

`ip_api` consumes a vector of IP addresses and geolocates them via ip-api.com.

Usage

```
ip_api(ip_addresses, as_data_frame = TRUE, delay = FALSE)
```

Arguments

`ip_addresses` a character vector of IP addresses

`as_data_frame` whether to return the results as a data.frame or not. Set to TRUE by default.

`delay` whether or not to delay each request by 400ms. ip-api.com has a maximum threshold of 150 requests a minute; if you're parallelising calls, you might run into this. `delay` allows you to set a delay between requests, taking advantage of parallelisation while avoiding running into this threshold. Set to FALSE by default

Value

either a data.frame or a list of vectors. If an IP cannot be geolocated, it will provide an error message: see the examples for field names and examples of each possible output.

See Also

ip_info and db_ip for other online geolocation APIs.

Examples

```
## Not run:
#Valid, data.frame output
result <- ip_api("2607:FB90:426:DC1D:CFC4:4875:8BC2:4D93")

#Invalid, data.frame output
result <- ip_api("argh")

#Valid list output
result <- ip_api("2607:FB90:426:DC1D:CFC4:4875:8BC2:4D93", as_data_frame = FALSE)

#Invalid list output
result <- ip_api("argh", as_data_frame = FALSE)

## End(Not run)
```

---

ip_info *Geolocate IP Addresses Through ipinfo.io*

Description

ip_info consumes a vector of IP addresses and geolocates them via ipinfo.io.

Usage

```
ip_info(ip_addresses, token = NULL)
```

Arguments

ip_addresses	a character vector of IP addresses
token	optionally, an API token. If you don't use one, you can still use the system, but requests will be capped to 1,000 a day.

Value

either a data.frame containing the geolocated information. If an IP cannot be geolocated, or values are not available, the fields will be filled with NA values.

See Also

ip_api and db_ip for other online geolocation APIs.

Examples

```
## Not run:
#Valid, data.frame output
result <- ip_info("2607:FB90:426:DC1D:CFC4:4875:8BC2:4D93")

#Invalid, data.frame output
result <- ip_info("argh")

#Valid list output
result <- ip_info("2607:FB90:426:DC1D:CFC4:4875:8BC2:4D93", as_data_frame = FALSE)

#Invalid list output
result <- ip_info("argh", as_data_frame = FALSE)

## End(Not run)
```

---

maxmind *Geolocate IP Addresses through MaxMind Databases*

Description

MaxMind does a set of proprietary geolocation databases - they're pretty accurate! maxmind provides a connector to MaxMind services.

Usage

```
maxmind(
  ips,
  file,
  fields = c("continent_name", "country_name", "country_code")
)
```

Arguments

ips	a character vector of IP addresses (IPv4 and IPv6 both work)
file	the full path to the .mmdb file you want to query.
fields	the fields you want to retrieve - a vector of any combination of:

- continent_name: the English-language name of the continent. Requires a country or city database.
- country_name: the English-language name of the country. Requires a country or city database.
- country_code: the ISO code of the country. Requires a country or city database.

- region_name: the English-language name of the region. Requires a city database.
- city_name: the English-language name of the city. Requires a city database.
- postcode: The approximate post/ZIP code. Requires a city database.
- city_geoname_id: a unique ID representing a city. Requires a city database.
- timezone: the tzdata-compatible time zone. Requires a city database.
- longitude: longitude of location. Requires a city database.
- latitude: latitude of location. Requires a city database.
- isp: name of ISP. Requires an ISP database.
- organization: name of organization. Requires an ISP database.
- asn: Autonomous System Number. Requires an ISP database.
- aso: Autonomous System Organization. Requires an ISP database.
- connection: the type of internet connection. Requires a connection type/netspeed database.

Details

`geolookup` uses the MaxMind GeoIP2 databases to geolocate IP addresses, retrieving any of the data listed in `fields`. Different fields are appropriate for different provided files; the connection type databases, for example, contain connection types and nothing else, while the city- and country-level files don't contain connection types at all.

`rgeolocate` ships with a country-level database (accessing it can be seen in the examples). If you need city-level data, or other MaxMind databases, you'll need to download the `.mmdb` files yourself - for CRAN and/or copyright reasons, depending, we cannot include them.

In the event that the file provided does not have the field you have requested (or the IP address does not have an entry for that field), NA will be returned instead. In the event that the IP address doesn't have an entry in the file at all, NA will be returned for every field.

Examples

```
# An example, using the country-level dataset shipped with rgeolocate.
file <- system.file("extdata","GeoLite2-Country.mmdb", package = "rgeolocate")
results <- maxmind("196.200.60.51", file, "country_code")
```

`rgeolocate` *IP Geolocation in R*

Description

This package aims to provide connectors to myriad online and offline sources for taking IP addresses and geolocating them to country, city, timezone and a whole other host of goodies. For individual connectors, see the package index. It depends on libmaxminddb, which can be obtained and installed from https://github.com/maxmind/libmaxminddb

Index