Package 'amapR'

July 4, 2021

Type Package

Title An R package using AMap Web Service API to convert between addresses and coordinates

Version 0.2.6

Author Xiaojun Lin <linxiaojun@scu.edu.cn>

Maintainer Xiaojun Lin <linxiaojun@scu.edu.cn>

Description

This R package provides useful tools for research using AMap Web Service API. 'amapR' currently can be use to convert between Chinese addresses and coordinates. Benefiting from the R parallel computing, this package has speed advantage in handling large data of addresses or coordinates. Theoratically, the more CPU cores you have, the faster the functions in this package will be. Please note that the Amap Web Service API have set the query limit for developers (e.g., the upper query limits for personal certified developer are 200 times per second and 3 million times per day).

URL https://github.com/xiaojunlin/amapR

BugReports https://github.com/xiaojunlin/amapR/issues

Imports data.table, sf, jsonlite, progress, parallel, doSNOW, foreach, stringr, stats, utils

NeedsCompilation no

Depends R (>= 4.0.0), dplyr

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

R topics documented:

geocoord 2
geolocation 3
getmap 4
transcoord 5

Index **7**

geocoord *Convert addresses into coordinates*

Description

Convert addresses into coordinates

Usage

```
geocoord(data, address, city = "", ncore = 999, nquery = 10)
```

Arguments

data	The dataset, a data.frame or data.table
address	The column name of address
city	Specify the city to query. This argument supports the city name in Chinese, the city name in pinyin, the administrative code of city or the city code defined by Amap. By default, this argument is empty. For more information, see the Amap official documents at https://lbs.amap.com/api/webservice/guide/api/georegeo.
ncore	The specific number of CPU cores used (ncore = 999 by default, which indicates the maximum of CPU cores minus 1 were used in parallel computing if your CPU is less than 999 cores)
nquery	The number of query in each batch (nquery = 10 by default). This argument is used to avoid the http 413 error when the request url is too long.

Value

a data.table which adds the formatted address, longitude and latitude in the original data set.

Note

According to the official document of AMap Web Service API, the address in the data set should be in Chinese format. If a address is in English or includes special characters (i.e., ?, -, >, _, etc.), the function may return empty result for this address automatically.

References

Amap. Official documents for developers: Web Service API. https://lbs.amap.com/api/webservice/summary

Examples

```
## Not run:
library(amapR)
options(amap.key = "xxxxxxxxxxx")

# Note: The "address" is the column having Chinese addresses, and the data set named "test"
# should be a data.frame or a data.table.
results <- geocoord(data = test, address = "address")

# Set the specific number of CPU cores used and the number of query in each batch
results <- geocoord(data = test, address = "address", ncore = 4, nquery = 5)
```

```
# Specify the city to query
results <- geocoord(data = test, address = "address", city = "chengdu")

## End(Not run)
```

`geolocation` *Convert the coordinates into formatted addresses*

Description

Convert the coordinates into formatted addresses

Usage

```
geolocation(data, longitude, latitude, ncore = 999)
```

Arguments

`data`	The dataset, a data.frame or data.table
`longitude`	The column having longitude
`latitude`	The column having latitude
`ncore`	The specific number of CPU cores used (ncore = 999 by default, which indicates the maximum of CPU cores minus 1 were used in parallel computing if your CPU is less than 999 cores)

Value

a data.table which adds the formatted address in the original data set.

Note

The value of "longitude" or "latitude" should be digits in numeric or character format. If not, the function may return empty result for this coordinate automatically.

References

Amap. Official documents for developers: Web Service API. https://lbs.amap.com/api/webservice/summary

Examples

```
## Not run:
library(amapR)
options(amap.key = "xxxxxxxxxxxxxxx")

# Completed data
test <- data.frame(n = 1:5000, lng = c(114.4345,104.0837), lat = c(30.51105, 30.63087))
results <- geolocation(data = test, longitude = "lng", latitude = "lat")

# When the column 'lng' has missing value
test <- data.frame(n = 1:5000, lng = c(114.4345,''), lat = c(30.51105, 30.63087))
```

```
results <- geolocation(data = test, longitude = "lng", latitude = "lat")

# When the column 'lng' has special characters
test <- data.frame(n = 1:5000, lng = c(114.4345,'?'), lat = c(30.51105, 30.63087))
results <- geolocation(data = test, longitude = "lng", latitude = "lat")

## End(Not run)
```

---

getmap *Get the map data with sf format*

Description

Get the map data at province, city or district level

Usage

```
getmap(adcode, level = "default")
```

Arguments

adcode	The administrative code
level	The level of map data, including province, city and district

Value

sf data

Examples

```
## Not run:
library(ggplot2)
# China map
# (1) China map at province level
china <- getmap(adcode = 100000, level = "province")
ggplot(china) +
  geom_sf(fill = "white") +
  theme_bw()
# (2) China map at city level
china <- getmap(adcode = 100000, level = "city")
ggplot(china) +
  geom_sf(fill = "white") +
  theme_bw()
# (3) China map at district level
china <- getmap(adcode = 100000, level = "district")
ggplot(china) +
  geom_sf(fill = "white") +
  theme_bw()
# (4) missing value in the argment "level"
china4 <- getmap(adcode = 100000)
ggplot(china) +
  geom_sf(fill = "white") +
  theme_bw()
```

```
# Sichuan
# (1) map at province level
sichuan <- getmap(adcode = 510000, level = "province")
ggplot(sichuan) +
  geom_sf(fill = "white") +
  theme_bw()
# (2) map at city level
sichuan <- getmap(adcode = 510000, level = "city")
ggplot(sichuan) +
  geom_sf(fill = "white") +
  theme_bw()
# (3) map at district level
sichuan <- getmap(adcode = 510000, level = "district")
ggplot(sichuan) +
  geom_sf(fill = "white") +
  theme_bw()

# Chengdu
# (1) map at city level
chengdu <- getmap(adcode = 510100, level = "city")
ggplot(chengdu) +
  geom_sf(fill = "white") +
  theme_bw()
# (2) fetch and map at district level
chengdu <- getmap(adcode = 510100, level = "district")
ggplot(chengdu) +
  geom_sf(fill = "white") +
  theme_bw()

## End(Not run)
```

`transcoord` *Transform the coordinates from other coordinate systems to Amap system*

Description

This function supports to transform the coordinates from three other coordinate systems (including baidu, GPS and mapbar) to Amap system

Usage

```
transcoord(data, longitude, latitude, coordsys = "autonavi", ncore = 999)
```

Arguments

`data`	The dataset, a data.frame or data.table
`longitude`	The column having longitude
`latitude`	The column having latitude
`coordsys`	The coordinate system of your original location data, such as "gps", "baidu", "mapbar" and "autonavi" (coordsys = "autonavi" by default)

ncore	The specific number of CPU cores used (ncore = 999 by default, which indicates the maximum of CPU cores minus 1 were used in parallel computing if your CPU is less than 999 cores)

Value

a data.table which adds the transformed longitude and latitude using Amap System in the original data set

Note

The value of "longitude" or "latitude" should be digits in numeric or character format. If not, the function may return empty result for this coordinate automatically

References

Amap. Official documents for developers: Web Service API. https://lbs.amap.com/api/webservice/summary

Examples

```
## Not run:
library(amapR)
options(amap.key = "xxxxxxxxxxxxxxx")

# Completed data
test <- data.frame(n = 1:500, lng = c(114.4345,104.0837), lat = c(30.51105, 30.63087))
results <- transcoord(data = test, longitude = "lng", latitude = "lat", coordsys = "baidu")

## End(Not run)
```

Index

geocoord, 2
geolocation, 3
getmap, 4

transcoord, 5