# Package 'PostcodesioR'

June 30, 2017

**Type** Package

**Title** API wrapper around postcodes.io (free UK postcode lookup and geocoder)

**Version** 0.1.1

**Author** Eryk Walczak <eryk.j.walczak@gmail.com>

**Maintainer** Eryk Walczak <eryk.j.walczak@gmail.com>

**Description** Free UK geocoding using data from Office for National Statistics. It is using several functions to get information about post codes, outward codes, reverse geocoding, nearest post codes/outward codes, validation, or randomly generate a post code.

**License** GPL-3 + file LICENSE

**URL** https://github.com/erzk/PostcodesioR

**LazyData** TRUE

**Depends** R (>= 3.1),
httr

**RoxygenNote** 6.0.1

**Suggests** knitr,
rmarkdown,
testthat,
covr

**VignetteBuilder** knitr

## R topics documented:

---

`bulk_postcode_lookup` *Bulk postcode lookup*

---

## Description

Returns a list of matching postcodes and respective available data.

## Usage

```
bulk_postcode_lookup(postcodes)
```

## Arguments

| | |
|---|---|
| `postcodes` | Accepts a list of postcodes. Accepts up to 100 postcodes. |

## Value

A list.

## Examples

```
pc_list <- list(postcodes = c("PR3 0SG", "M45 6GN", "EX165BL"))
bulk_postcode_lookup(pc_list)
```

---

`bulk_reverse_geocoding` *Reverse geocoding*

---

## Description

Returns nearest postcodes for a given longitude and latitude. Accepts up to 100 geolocations.

## Usage

```
bulk_reverse_geocoding(geolocations)
```

## Arguments

| | |
|---|---|
| `geolocations` | A list containing an array of geolocation objects. |

## Details

This method requires a JSON object containing an array of geolocation objects to be POSTed. Each geolocation object accepts an optional radius (meters) and limit argument. Both default to 100m and 10 respectively. It also accepts a wideSearch argument.

## Value

A list with the geocoded data.

## Examples

```
geolocations_list <- structure(
list(
geolocations = structure(
list(
longitude = c(-3.15807731271522, -1.12935802905177),
latitude = c(51.4799900627036, 50.7186356978817),
limit = c(NA, 100L),
radius = c(NA, 500L)),
.Names = c("longitude", "latitude", "limit", "radius"),
class = "data.frame",
row.names = 1:2)),
.Names = "geolocations")

bulk_reverse_geocoding(geolocations_list)
```

---

# nearest_outcode *Nearest outcode*

---

## Description

Returns nearest outcodes for a given outcode.

## Usage

```
nearest_outcode(outcode, limit = 10, radius = 5000)
```

## Arguments

| | |
|---|---|
| outcode | A string with a UK postcode. |
| limit | An integer. Optional parameter. Limits number of postcodes matches to return. Defaults to 10. Needs to be less than 100. |
| radius | An integer. Optional parameter. Limits number of postcodes matches to return. Defaults to 5,000m. Needs to be less than 25,000m. |

## Value

A list of geographical properties.

## Examples

```
nearest_outcode("EC1Y")
nearest_outcode("EC1Y", limit = 11)
nearest_outcode("EC1Y", limit = 11, radius = 6000)
```

## nearest_postcode *Nearest postcode*

### Description

Returns nearest postcodes for a given postcode.

### Usage

```
nearest_postcode(postcode, limit = 10, radius = 100)
```

### Arguments

| | |
|---|---|
| postcode | A string. Valid UK postcode. |
| limit | A string or integer. Limits number of postcodes matches to return. Defaults to 10. Needs to be lower than 100. |
| radius | Limits number of postcodes matches to return. Defaults to 100m. Needs to be less than 2,000m. |

### Value

A list of geographic properties of the nearest postcode.

### Examples

```
nearest_postcode("EC1Y 8LX")
nearest_postcode("EC1Y 8LX", limit = 11)
nearest_postcode("EC1Y 8LX", limit = 12, radius = 200)
```

## outcode_reverse_geocoding *Outcode reverse geocoding*

### Description

Returns nearest outcodes for a given longitude and latitude.

### Usage

```
outcode_reverse_geocoding(longitude, latitude, limit = 10, radius = 5000)
```

### Arguments

| | |
|---|---|
| longitude | A string, integer or float. Needs to have at least two decimal points. |
| latitude | A string, integer or float. Needs to have at least two decimal points. |
| limit | A string, integer or float. Limits number of postcodes matches to return. Defaults to 10. Needs to be less than 100. |
| radius | A string, integer or float. Limits number of postcodes matches to return. Defaults to 5,000m. Needs to be less than 25,000m. |

## Value

A list of geographical properties.

## Examples

```
outcode_reverse_geocoding("-3.15", "51.47")
outcode_reverse_geocoding(-3.15, 51.47)
outcode_reverse_geocoding("-3.15807731271522", "51.4799900627036")
outcode_reverse_geocoding(-3.15, 51.47, limit = 11, radius = 20000)
```

---

# outward_code_lookup *Outward code lookup*

---

## Description

Geolocation data for the centroid of the outward code specified.

## Usage

```
outward_code_lookup(outcode)
```

## Arguments

| | |
|---|---|
| outcode | A string. The outward code representing the first half of any postcode (separated by a space). |

## Value

The list of geographical properties.

## Examples

```
outward_code_lookup("E1")
```

---

# place_lookup *Place Lookup*

---

## Description

Lookup a place by code. Returns all available data if found. Returns 404 if place does not exist.

## Usage

```
place_lookup(code)
```

## Arguments

| | |
|---|---|
| code | A string. The unique identifier for places - Ordnance Survey (OSGB) code. |

## Value

A list with available places.

## Examples

```
place_lookup("osgb4000000074544700")
```

---

`place_query` *Place Query*

---

## Description

Submit a place query and receive a complete list of places matches and associated data.

## Usage

```
place_query(place, limit = 10)
```

## Arguments

| | |
|---|---|
| `place` | A string. Name of a place to search for. |
| `limit` | An integer. Limits the number of matches to return. Defaults to 10. Needs to be less than 100. |

## Value

A list with available places.

## Examples

```
place_query("Hills")
place_query("Hills", limit = 12)
```

---

`postcode_autocomplete` *Postcode autocomplete*

---

## Description

Convenience method to return an list of matching postcodes.

## Usage

```
postcode_autocomplete(postcode, limit = 10)
```

## Arguments

| | |
|---|---|
| `postcode` | A string. Valid UK postcode. |
| `limit` | An integer. Limits number of postcodes matches to return. Defaults to 10. Needs to be less than 100. |

### Value

A list of suggested postcodes.

### Examples

```
postcode_autocomplete("E1")
postcode_autocomplete("E1", limit = 11)
```

---

## postcode_lookup *Postcode lookup*

---

### Description

Lookup a postcode.

### Usage

```
postcode_lookup(postcode)
```

### Arguments

| | |
|---|---|
| postcode | A string. Valid UK postcode. |

### Value

A list. Returns all available data if found. Returns 404 if postcode does not exist.

### Examples

```
postcode_lookup("EC1Y8LX")
postcode_lookup("EC1Y 8LX")
```

---

## postcode_query *Postcode query*

---

### Description

Submit a postcode query and receive a complete list of postcode matches and all associated postcode data.

### Usage

```
postcode_query(postcode, limit = 10)
```

### Arguments

| | |
|---|---|
| postcode | A string. Valid UK postcode. |
| limit | An integer. Limits the number of matches to return. Defaults to 10. Needs to be less than 100. |

## Value

A list of geographic properties.

## Examples

```
postcode_query("EC1Y8LX")
postcode_query("EC1", limit = 11)
```

---

`postcode_validation` *Postcode validation*

---

## Description

Convenience method to validate a postcode.

## Usage

```
postcode_validation(postcode)
```

## Arguments

| | |
|---|---|
| `postcode` | A string. Valid UK postcode. |

## Value

A logical vector: True or False (meaning respectively valid or invalid postcode).

## Examples

```
postcode_validation("EC1Y 8LX") # returns TRUE
postcode_validation("XYZ") # returns FALSE
```

---

`random_place` *Random Place*

---

## Description

Returns a random place and all associated data

## Usage

```
random_place()
```

## Value

A data frame describing a random place and all associated data.

## Examples

```
random_place()
```

## random_postcode *Random postcode*

### Description

Returns a random postcode and all available data for that postcode.

### Usage

```
random_postcode(outcode = NULL)
```

### Arguments

| | |
|---|---|
| outcode | A string. Filters random postcodes by outcode. Returns null if invalid outcode. Optional. |

### Value

A data frame with a random post code with corresponding characteristics.

### Examples

```
random_postcode()
random_postcode("N1")
```

## reverse_geocoding *Reverse geocoding*

### Description

Returns nearest postcodes for a given longitude and latitude.

### Usage

```
reverse_geocoding(longitude, latitude, limit = 10, radius = 100,
  wideSearch = NULL)
```

### Arguments

| | |
|---|---|
| longitude | A string. Needs to have at least three decimal points. |
| latitude | A string. Needs to have at least three decimal points. |
| limit | An integer. Limits number of postcodes matches to return. Defaults to 10. Needs to be less than 100. |
| radius | An integer. Limits number of postcodes matches to return. Defaults to 100m. Needs to be less than 2,000m. |
| wideSearch | TRUE or FALSE. Search up to 20km radius, but subject to a maximum of 10 results. Since lookups over a wide area can be very expensive, we've created this method to allow you choose to make the trade off between search radius and number of results. Defaults to false. When enabled, radius and limits over 10 are ignored. |

## Value

A list with available data.

## Examples

```
reverse_geocoding(0.127, 51.507)
reverse_geocoding("0.1275", "51.5073", limit = 3)
reverse_geocoding("0.1275", "51.5073", limit = 11, radius = 200)
```

# Index