

# Package ‘rgugik’

April 11, 2021

**Type** Package

**Title** Search and Retrieve Spatial Data from 'GUGiK'

**Version** 0.3.0

**Description** Automatic open data acquisition from resources of Polish Head Office of Geodesy and Cartography ('Główny Urząd Geodezji i Kartografii') (<[www.gugik.gov.pl](http://www.gugik.gov.pl)>).

Available datasets include various types of numeric, raster and vector data, such as orthophotomaps, digital elevation models (digital terrain models, digital surface model, point clouds), state register of borders, spatial databases, geometries of cadastral parcels, 3D models of buildings, and more. It is also possible to geocode addresses or objects using the `geocodePL_get()` function.

**License** MIT + file LICENSE

**Depends** R (>= 3.5)

**Imports** sf, jsonlite, openssl

**Suggests** curl, knitr, rmarkdown, tibble, testthat, stars

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**URL** <https://kadyb.github.io/rgugik/>, <https://github.com/kadyb/rgugik>

**BugReports** <https://github.com/kadyb/rgugik/issues>

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Krzysztof Dyba [aut, cre] (<<https://orcid.org/0000-0002-8614-3816>>),  
Jakub Nowosad [aut] (<<https://orcid.org/0000-0002-1057-3721>>),  
Maciej Beręsewicz [ctb] (<<https://orcid.org/0000-0002-8281-4301>>),  
GUGiK [ctb] (source of the data)

**Maintainer** Krzysztof Dyba <[adres7@gmail.com](mailto:adres7@gmail.com)>

**Repository** CRAN

**Date/Publication** 2021-04-11 18:00:02 UTC

**R topics documented:**

borders_download . . . . .	2
borders_get . . . . .	3
commune_names . . . . .	4
county_names . . . . .	4
DEM_request . . . . .	5
emuia_download . . . . .	5
geocodePL_get . . . . .	6
geodb_download . . . . .	7
geonames_download . . . . .	8
minmaxDTM_get . . . . .	9
models3D_download . . . . .	10
ortho_request . . . . .	11
parcel_get . . . . .	11
pointDTM100_download . . . . .	12
pointDTM_get . . . . .	13
tile_download . . . . .	14
topodb_download . . . . .	15
voivodeship_names . . . . .	16
<b>Index</b>	<b>17</b>

---

borders_download	<i>Download State Register of Borders</i>
------------------	---

---

**Description**

Download State Register of Borders

**Usage**

```
borders_download(type, outdir = ".", unzip = TRUE, ...)
```

**Arguments**

type	"administrative units", "special units" or "addresses"
outdir	(optional) name of the output directory; by default, files are saved in the working directory
unzip	TRUE (default) or FALSE, when TRUE the downloaded archive will be extracted and removed
...	additional argument for <code>utils::download.file()</code>

**Value**

a selected data type in SHP format

## Examples

```
## Not run:  
borders_download("administrative units") # 375 MB  
  
## End(Not run)
```

---

borders_get	<i>Get the boundaries of administrative units</i>
-------------	---

---

## Description

Get the boundaries of administrative units

## Usage

```
borders_get(voivodeship = NULL, county = NULL, commune = NULL, TERYT = NULL)
```

## Arguments

voivodeship	selected voivodeships in Polish. Check <a href="#">voivodeship_names()</a> function
county	county names in Polish. Check <a href="#">county_names()</a> function
commune	commune names in Polish. Check <a href="#">commune_names()</a> function
TERYT	voivodeships, counties or communes (2, 4 or 7 characters)

## Value

a sf data.frame (EPSG: 2180)

## Examples

```
## Not run:  
voivodeship_geom = borders_get(voivodeship = "lubuskie") # 494 KB  
county_geom = borders_get(county = "Sopot") # 18 KB  
commune_geom = borders_get(commune = c("Hel", "Krynica Morska")) # 11 KB  
  
## End(Not run)
```

---

`commune_names`*Communes in Poland*

---

**Description**

The data frame contains names of communes, and their identifiers (TERC, 7 characters).

**Usage**`commune_names`**Format**

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 2477 rows and 2 columns.

**Examples**`commune_names`

---

`county_names`*Counties in Poland*

---

**Description**

The data frame contains the names of counties, their identifiers (TERYT, 4 characters) and the availability of building models in the LOD2 standard (logical value).

**Usage**`county_names`**Format**

An object of class `data.frame` with 380 rows and 3 columns.

**Examples**`county_names`

---

DEM_request	<i>Get metadata and links to available digital elevation models</i>
-------------	---

---

**Description**

Get metadata and links to available digital elevation models

**Usage**

```
DEM_request(x)
```

**Arguments**

x                    an sf/sfc object with one or more features (requests are based on the bounding boxes of the provided features)

**Value**

a data frame with metadata and links to the digital elevation models (different formats of digital terrain model, digital surface model and point clouds)

**Examples**

```
## Not run:
library(sf)
polygon_path = system.file("datasets/search_area.gpkg", package = "rgugik")
polygon = read_sf(polygon_path)
req_df = DEM_request(polygon)

# simple filtering by attributes
req_df = req_df[req_df$year > 2018, ]
req_df = req_df[req_df$product == "PointCloud" & req_df$format == "LAS", ]

## End(Not run)
```

---

emuia_download	<i>Download Register of Towns, Streets and Addresses for communes</i>
----------------	---

---

**Description**

Download Register of Towns, Streets and Addresses for communes

**Usage**

```
emuia_download(commune = NULL, TERYT = NULL, outdir = ".", unzip = TRUE, ...)
```

**Arguments**

commune	commune name in Polish. Check <a href="#">commune_names()</a> function.
TERYT	county ID (7 characters)
outdir	(optional) name of the output directory; by default, files are saved in the working directory
unzip	TRUE (default) or FALSE, when TRUE the downloaded archive will be extracted and removed
...	additional argument for <a href="#">utils::download.file()</a>

**Value**

a register in SHP format

**Examples**

```
## Not run:
emuia_download(commune = "Kotla") # 38 KB
emuia_download(TERYT = c("0203042", "2412032")) # 75 KB

## End(Not run)
```

---

geocodePL\_get

---

*Convert addresses and objects to geographic coordinates*


---

**Description**

Convert addresses and objects to geographic coordinates

**Usage**

```
geocodePL_get(
  address = NULL,
  road = NULL,
  rail_crossing = NULL,
  geoname = NULL
)
```

**Arguments**

address	place with or without street and house number
road	road number with or without mileage
rail_crossing	rail crossing identifier (11 characters including 2 spaces, format: "XXX XXX XXX")
geoname	name of the geographical object from State Register of Geographical Names (function <a href="#">geonames_download()</a> )

**Value**

a sf data.frame (EPSG: 2180) with metadata

**Examples**

```
## Not run:
geocodePL_get(address = "Marki") # place
geocodePL_get(address = "Marki, Andersa") # place and street
geocodePL_get(address = "Marki, Andersa 1") # place, street and house number
geocodePL_get(address = "Królewskie Brzeziny 13") # place and house number

geocodePL_get(road = "632") # road number
geocodePL_get(road = "632 55") # road number and mileage

geocodePL_get(rail_crossing = "001 018 478")

geocodePL_get(geoname = "Las Mierzei") # physiographic object

## End(Not run)
```

---

geodb\_download

*Download General Geographic Databases for entire voivodeships*

---

**Description**

Download General Geographic Databases for entire voivodeships

**Usage**

```
geodb_download(voivodeships, outdir = ".", unzip = TRUE, ...)
```

**Arguments**

voivodeships	selected voivodeships in Polish or English, or TERC (object <a href="#">voivodeship_names</a> can be helpful)
outdir	(optional) name of the output directory; by default, files are saved in the working directory
unzip	TRUE (default) or FALSE, when TRUE the downloaded archive will be extracted and removed
...	additional argument for <a href="#">utils::download.file()</a>

**Value**

a database in Geography Markup Language format (.GML), the content and detail level corresponds to the general geographic map in the scale of 1:250000

## References

description of topographical and general geographical databases, and technical standards for making maps (in Polish): [http://www.gugik.gov.pl/\\_\\_data/assets/pdf\\_file/0005/208661/rozp\\_BDOT10k\\_BD00.pdf](http://www.gugik.gov.pl/__data/assets/pdf_file/0005/208661/rozp_BDOT10k_BD00.pdf)

brief description of categories and layer names (in English and Polish): [https://kadyb.github.io/rgugik/articles/articles/spatialdb\\_description.html](https://kadyb.github.io/rgugik/articles/articles/spatialdb_description.html)

## Examples

```
## Not run:
geodb_download(c("opolskie", "lubuskie")) # 12.7 MB
geodb_download(c("Opole", "Lubusz")) # 12.7 MB
geodb_download(c("16", "08")) # 12.7 MB

## End(Not run)
```

---

geonames_download	<i>Download State Register of Geographical Names</i>
-------------------	--

---

## Description

Download State Register of Geographical Names

## Usage

```
geonames_download(type, format = "SHP", outdir = ".", unzip = TRUE, ...)
```

## Arguments

type	names of places ("place") and/or physiographic objects ("object")
format	data format ("GML", "SHP" (default) and/or "XLSX")
outdir	(optional) name of the output directory; by default, files are saved in the working directory
unzip	TRUE (default) or FALSE, when TRUE the downloaded archive will be extracted and removed
...	additional argument for <a href="#">utils::download.file()</a>

## Value

a selected data type in the specified format

## References

<http://isap.sejm.gov.pl/isap.nsf/download.xsp/WDU2015000219/O/D20150219.pdf>



## Examples

```
## Not run:  
geonames_download(type = "place", format = "SHP") # 18.2 MB  
  
## End(Not run)
```

---

minmaxDTM\_get

*Get minimum and maximum elevation for a given polygon*

---

## Description

Get minimum and maximum elevation for a given polygon

## Usage

```
minmaxDTM_get(polygon)
```

## Arguments

polygon	the polygon layer with only one object (area less than 10 ha), the larger the polygon area, the lower DTM resolution, the input coordinate system must be EPSG:2180
---------	---

## Value

a data frame with vector points and min/max terrain elevation (EPSG:2180)

## Examples

```
## Not run:  
library(sf)  
polygon_path = system.file("datasets/search_area.gpkg", package = "rgugik")  
polygon = read_sf(polygon_path)  
minmax = minmaxDTM_get(polygon)  
  
## End(Not run)
```

---

models3D_download	<i>Download 3D models of buildings for counties</i>
-------------------	---

---

## Description

Download 3D models of buildings for counties

## Usage

```
models3D_download(  
  county = NULL,  
  TERYT = NULL,  
  LOD = "LOD1",  
  outdir = ".",  
  unzip = TRUE,  
  ...  
)
```

## Arguments

county	county name in Polish. Check <a href="#">county_names()</a> function.
TERYT	county ID (4 characters)
LOD	level of detail for building models ("LOD1" or "LOD2"). "LOD1" is default. "LOD2" is only available for ten voivodeships (TERC: "04", "06", "12", "14", "16", "18", "20", "24", "26", "28"). Check <a href="#">voivodeship_names()</a> function.
outdir	(optional) name of the output directory; by default, files are saved in the working directory
unzip	TRUE (default) or FALSE, when TRUE the downloaded archive will be extracted and removed
...	additional argument for <a href="#">utils::download.file()</a>

## Value

models of buildings in Geography Markup Language format (.GML)

## Examples

```
## Not run:  
models3D_download(TERYT = c("2476", "2264")) # 3.6 MB  
models3D_download(county = "sejneński", LOD = "LOD2") # 7.0 MB  
  
## End(Not run)
```

---

ortho_request	<i>Get metadata and links to available orthoimages</i>
---------------	--

---

**Description**

Get metadata and links to available orthoimages

**Usage**

```
ortho_request(x)
```

```
orto_request(x)
```

**Arguments**

x                    an sf/sfc object with one or more features (requests are based on the bounding boxes of the provided features)

**Value**

a data frame with metadata and links to the orthoimages

**Examples**

```
## Not run:
library(sf)
polygon_path = system.file("datasets/search_area.gpkg", package = "rgugik")
polygon = read_sf(polygon_path)
req_df = ortho_request(polygon)

# simple filtering by attributes
req_df = req_df[req_df$composition == "CIR", ]
req_df = req_df[req_df$resolution <= 0.25 & req_df$year >= 2016, ]

## End(Not run)
```

---

parcel_get	<i>Get the geometry of cadastral parcels</i>
------------	--

---

**Description**

Get the geometry of cadastral parcels

**Usage**

```
parcel_get(TERYT = NULL, X = NULL, Y = NULL)
```

**Arguments**

TERYT	parcel ID (18 characters, e.g. "141201_1.0001.6509")
X	longitude (EPSG: 2180)
Y	latitude (EPSG: 2180)

**Value**

a simple feature geometry (in case of TERYT) or data frame with simple feature geometry and TERYT (in case of coordinates)

**Examples**

```
## Not run:
parcel = parcel_get(TERYT = "141201_1.0001.6509")
parcel = parcel_get(X = 313380.5, Y = 460166.4)

## End(Not run)
```

---

pointDTM100\_download *Download digital terrain models for voivodeships (100 m resolution)*

---

**Description**

Download digital terrain models for voivodeships (100 m resolution)

**Usage**

```
pointDTM100_download(voivodeships, outdir = ".", unzip = TRUE, ...)
```

**Arguments**

voivodeships	selected voivodeships in Polish or English, or TERC (function <a href="#">voivodeship_names()</a> can be helpful)
outdir	(optional) name of the output directory; by default, files are saved in the working directory
unzip	TRUE (default) or FALSE, when TRUE the downloaded archive will be extracted and removed
...	additional argument for <a href="#">utils::download.file()</a>

**Value**

text files with X, Y, Z columns (EPSG:2180)

**Examples**

```
## Not run:
pointDTM100_download(c("opolskie", "świętokrzyskie")) # 8.5 MB
pointDTM100_download(c("Opole", "Swietokrzyskie")) # 8.5 MB
pointDTM100_download(c("16", "26")) # 8.5 MB

## End(Not run)
```

---

pointDTM_get	<i>Get terrain elevation for a given polygon</i>
--------------	--

---

**Description**

Get terrain elevation for a given polygon

**Usage**

```
pointDTM_get(polygon, distance = 1, print_iter = TRUE)
```

**Arguments**

polygon	the polygon layer with only one object (its area is limited to the 20 ha * distance parameter), the input coordinate system must be EPSG:2180
distance	distance between points in meters (must be integer and greater than 1)
print_iter	print the current iteration of all (logical, TRUE default)

**Value**

a data frame with vector points and terrain elevation (EPSG:2180, Vertical Reference System:PL-KRON86-NH)

**Examples**

```
## Not run:
library(sf)
polygon_path = system.file("datasets/search_area.gpkg", package = "rgugik")
polygon = read_sf(polygon_path)
DTM = pointDTM_get(polygon, distance = 2)

## End(Not run)
```

---

tile_download	<i>Download requested tiles</i>
---------------	---------------------------------

---

### Description

Download requested tiles

### Usage

```
tile_download(
  df_req,
  outdir = ".",
  unzip = TRUE,
  check_SHA = FALSE,
  print_iter = TRUE,
  ...
)
```

### Arguments

df_req	a data frame obtained using the <a href="#">ortho_request()</a> and <a href="#">DEM_request()</a> functions
outdir	(optional) name of the output directory; by default, files are saved in the working directory
unzip	TRUE (default) or FALSE, when TRUE the downloaded archive will be extracted and removed; only suitable for certain elevation data
check_SHA	check the integrity of downloaded files (logical, FALSE default)
print_iter	print the current iteration of all (logical, TRUE default)
...	additional argument for <a href="#">utils::download.file()</a>

### Value

georeferenced tiles with properties (resolution, year, etc.) as specified in the input data frame

### Examples

```
## Not run:
library(sf)
polygon_path = system.file("datasets/search_area.gpkg", package = "rgugik")
polygon = read_sf(polygon_path)

req_df = ortho_request(polygon)
tile_download(req_df[1, ]) # download the first image only

req_df = DEM_request(polygon)
tile_download(req_df[1, ]) # download the first DEM only

## End(Not run)
```

---

topodb_download	<i>Download Topographic Databases for counties</i>
-----------------	--

---

## Description

Download Topographic Databases for counties

## Usage

```
topodb_download(county = NULL, TERYT = NULL, outdir = ".", unzip = TRUE, ...)
```

## Arguments

county	county name in Polish. Check <a href="#">county_names()</a> function.
TERYT	county ID (4 characters)
outdir	(optional) name of the output directory; by default, files are saved in the working directory
unzip	TRUE (default) or FALSE, when TRUE the downloaded archive will be extracted and removed
...	additional argument for <a href="#">utils::download.file()</a>

## Value

a database in Geography Markup Language format (.GML), the content and detail level corresponds to the topographic map in the scale of 1:10000

## References

description of topographical and general geographical databases, and technical standards for making maps (in Polish): [http://www.gugik.gov.pl/\\_\\_data/assets/pdf\\_file/0005/208661/rozp\\_BD0T10k\\_BD00.pdf](http://www.gugik.gov.pl/__data/assets/pdf_file/0005/208661/rozp_BD0T10k_BD00.pdf)

brief description of categories and layer names (in English and Polish): [https://kadyb.github.io/rgugik/articles/articles/spatialdb\\_description.html](https://kadyb.github.io/rgugik/articles/articles/spatialdb_description.html)

## Examples

```
## Not run:  
topodb_download(county = "Świętochłowice") # 2.4 MB  
topodb_download(TERYT = c("2476", "2264")) # 4.8 MB  
  
## End(Not run)
```

---

voivodeship_names	<i>Voivodeships in Poland</i>
-------------------	-------------------------------

---

**Description**

The data frame contains Polish and English names of voivodeships, and their identifiers (TERC, 2 characters).

**Usage**

```
voivodeship_names
```

**Format**

An object of class `data.frame` with 16 rows and 3 columns.

**Examples**

```
voivodeship_names
```



# Index

- \* **commune**
  - commune\_names, 4
- \* **county**
  - county\_names, 4
- \* **dataset**
  - commune\_names, 4
  - county\_names, 4
  - voivodeship\_names, 16
- \* **voivodeship**
  - voivodeship\_names, 16

borders\_download, 2  
borders\_get, 3

commune\_names, 4  
commune\_names(), 3, 6  
county\_names, 4  
county\_names(), 3, 10, 15

DEM\_request, 5  
DEM\_request(), 14

emuia\_download, 5

geocodePL\_get, 6  
geodb\_download, 7  
geonames\_download, 8  
geonames\_download(), 6

minmaxDTM\_get, 9  
models3D\_download, 10

ortho\_request, 11  
ortho\_request(), 14  
orto\_request(ortho\_request), 11

parcel\_get, 11  
pointDTM100\_download, 12  
pointDTM\_get, 13

tile\_download, 14

topodb\_download, 15

utils::download.file(), 2, 6–8, 10, 12, 14, 15

voivodeship\_names, 7, 16  
voivodeship\_names(), 3, 10, 12