Package 'CatastRo'

October 5, 2026

Title Interface to the Spanish 'Catastro' Web Services

Version 1.1.0

Description Access public spatial data from the Spanish 'Catastro' through its Infrastructure for Spatial Information in Europe ('INSPIRE') and related web services. Retrieve parcel, building, address and map image data and convert between property reference codes and coordinates.

License GPL-2

URL https://ropenspain.github.io/CatastRo/, https://github.com/rOpenSpain/CatastRo

BugReports https://github.com/rOpenSpain/CatastRo/issues

Depends R (>= 4.1.0)

Imports cli, dplyr, httr2 (>= 1.0.0), lifecycle, mapSpain (>= 1.0.0), rappdirs (>= 0.3.0), sf (>= 1.0.0), terra, tools, utils, xml2

Suggests ggplot2, knitr, quarto, testthat (>= 3.3.0), tibble, tidyterra, withr

VignetteBuilder quarto

Config/Needs/website ropenspain/rostemplate, devtools, sessioninfo, remotes, sfheaders, rapidjsonr, jsonify, geometries, magick

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Config/testthat/parallel true

Copyright Cadastral data © Dirección General del Catastro. See inst/COPYRIGHTS for data access and use conditions.

Encoding UTF-8

LazyData true

X-schema.org-isPartOf https://ropenspain.es/

X-schema.org-keywords catastro, cran, cran-r, gis, maps, r, r-package, ropenspain, rstats, spain, spatial, static-tiles

NeedsCompilation no

Author Ángel Delgado Panadero [aut, cph] (ORCID:
<https://orcid.org/0000-0002-8189-9251>),
Iñaki Ucar [ctb] (ORCID: <https://orcid.org/0000-0001-6403-5550>),
Diego Hernangómez [aut, cre] (ORCID:
<https://orcid.org/0000-0001-8457-4658>),
Dirección General del Catastro [dtc] (Provider of cadastral data and
web services)

Maintainer Diego Hernangómez <diego.hernangomezherrero@gmail.com>

Repository CRAN

Date/Publication 2026-10-05 10:10:02 UTC

Contents

catr_atom_get_address 2
catr_atom_get_address_db_all 4
catr_atom_get_buildings 5
catr_atom_get_buildings_db_all 7
catr_atom_get_parcels 9
catr_atom_get_parcels_db_all 10
catr_atom_search_munic 12
catr_clear_cache 13
catr_get_code_from_coords 14
catr_ovc_get_cod_munic 16
catr_ovc_get_cod_provinces 17
catr_ovc_get_cpmrc 18
catr_ovc_get_rccoor 20
catr_ovc_get_rccoor_distancia 21
catr_set_cache_dir 23
catr_srs_values 25
catr_wfs_get_address_bbox 26
catr_wfs_get_buildings_bbox 28
catr_wfs_get_parcels_bbox 30
catr_wms_get_layer 32
inspire_wfs_get 35

Index **37**

`catr_atom_get_address` *ATOM INSPIRE: Download all addresses for a municipality*

Description

Retrieve spatial data for all addresses in a municipality using the ATOM INSPIRE service. The result also contains the corresponding street information in fields prefixed with `tfname_*`.

Usage

```
catr_atom_get_address(
  munic,
  to = NULL,
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

munic	Municipality name, partial name or cadastral code. Use catr_atom_search_munic() to find cadastral codes.
to	Optional territorial office containing munic. Use this argument to narrow the search.
cache	**[Deprecated]** This argument is no longer supported because results are always cached.
update_cache	Logical. Whether to refresh the cached file. Defaults to FALSE.
cache_dir	Path to a cache directory. If NULL or FALSE, the function stores cached files in a temporary directory. See base::tempdir().
verbose	Logical. Whether to display informational messages.

Value

An sf object. Returns NULL if the data cannot be retrieved.

References

API documentation.

INSPIRE services for cadastral cartography.

See Also

Work with cadastral addresses: catr_atom_get_address_db_all(), catr_wfs_get_address_bbox()

Query ATOM INSPIRE services: catr_atom_get_address_db_all(), catr_atom_get_buildings(), catr_atom_get_buildings_db_all(), catr_atom_get_parcels(), catr_atom_get_parcels_db_all(), catr_atom_search_munic()

Examples

```
s <- catr_atom_get_address("Melque", to = "Segovia")

library(ggplot2)

ggplot(s) +
```

```
  geom_sf(aes(color = specification)) +
  coord_sf(
    xlim = c(376200, 376850),
    ylim = c(4545000, 4546000)
  ) +
  labs(
    title = "Addresses",
    subtitle = "Melque de Cercos, Segovia"
  )
```

`catr_atom_get_address_db_all`

ATOM INSPIRE: List address download URLs

Description

Create a table of URLs provided by the Spanish Cadastre ATOM INSPIRE service for downloading addresses.

`catr_atom_get_address_db_all()` provides a summary table with all territorial offices and their municipalities, excluding the Basque Country and Navarre. `catr_atom_get_address_db_to()` provides a table for one territorial office and its municipalities.

Usage

```
catr_atom_get_address_db_all(
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)

catr_atom_get_address_db_to(
  to,
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

`cache`	**[Deprecated]** This argument is no longer supported because results are always cached.
`update_cache`	Logical. Whether to refresh the cached file. Defaults to `FALSE`.
`cache_dir`	Path to a cache directory. If `NULL` or `FALSE`, the function stores cached files in a temporary directory. See `base::tempdir()`.

verbose	Logical. Whether to display informational messages.
to	Character string. Territorial office to match using base::grep().

Value

A tibble with the requested information. Returns NULL if the data cannot be retrieved or no territorial office matches to.

catr_atom_get_address_db_all() returns the following columns:

- territorial_office: Territorial office, corresponding to each province of Spain except the Basque Country and Navarre.
- url: ATOM URL for the corresponding territorial office.
- munic: Name of the municipality.
- date: Reference date of the data. The information from this service is updated twice a year.

catr_atom_get_address_db_to() returns munic, url and date for the selected territorial office.

Source

https://www.catastro.hacienda.gob.es/INSPIRE/Addresses/ES.SDGC.AD.atom.xml

See Also

Work with cadastral addresses: catr_atom_get_address(), catr_wfs_get_address_bbox()

Query ATOM INSPIRE services: catr_atom_get_address(), catr_atom_get_buildings(), catr_atom_get_buildings_db_all(), catr_atom_get_parcels(), catr_atom_get_parcels_db_all(), catr_atom_search_munic()

Examples

```
catr_atom_get_address_db_all()
```

---

catr_atom_get_buildings

ATOM INSPIRE: Download all buildings for a municipality

Description

Retrieve spatial data for all buildings in a municipality using the ATOM INSPIRE service.

Usage

```
catr_atom_get_buildings(
  munic,
  to = NULL,
  what = c("building", "buildingpart", "other"),
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

munic	Municipality name, partial name or cadastral code. Use catr_atom_search_munic() to find cadastral codes.
to	Optional territorial office containing munic. Use this argument to narrow the search.
what	Information to load, either "building" for buildings, "buildingpart" for parts of a building or "other" for other elements such as swimming pools.
cache	**[Deprecated]** This argument is no longer supported because results are always cached.
update_cache	Logical. Whether to refresh the cached file. Defaults to FALSE.
cache_dir	Path to a cache directory. If NULL or FALSE, the function stores cached files in a temporary directory. See base::tempdir().
verbose	Logical. Whether to display informational messages.

Value

An sf object. Returns NULL if the data cannot be retrieved.

References

API documentation.

INSPIRE services for cadastral cartography.

See Also

Work with cadastral buildings: catr_atom_get_buildings_db_all(), catr_wfs_get_buildings_bbox()

Query ATOM INSPIRE services: catr_atom_get_address(), catr_atom_get_address_db_all(), catr_atom_get_buildings_db_all(), catr_atom_get_parcels(), catr_atom_get_parcels_db_all(), catr_atom_search_munic()

Examples

```
s <- catr_atom_get_buildings("Nava de la Asuncion", to = "Segovia")

library(ggplot2)
ggplot(s) +
  geom_sf() +
  coord_sf(
    xlim = c(374500, 375500),
    ylim = c(4556500, 4557500)
  ) +
  labs(
    title = "Buildings",
    subtitle = "Nava de la Asuncion, Segovia"
  )
```

`catr_atom_get_buildings_db_all`
ATOM INSPIRE: List building download URLs

Description

Create a table of URLs provided by the Spanish Cadastre ATOM INSPIRE service for downloading buildings.

`catr_atom_get_buildings_db_all()` provides a summary table with all territorial offices and their municipalities, excluding the Basque Country and Navarre. `catr_atom_get_buildings_db_to()` provides a table for one territorial office and its municipalities.

Usage

```
catr_atom_get_buildings_db_all(
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)

catr_atom_get_buildings_db_to(
  to,
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

cache	**[Deprecated]** This argument is no longer supported because results are always cached.
update_cache	Logical. Whether to refresh the cached file. Defaults to FALSE.
cache_dir	Path to a cache directory. If NULL or FALSE, the function stores cached files in a temporary directory. See base::tempdir().
verbose	Logical. Whether to display informational messages.
to	Character string. Territorial office to match using base::grep().

Value

A tibble with the requested information. Returns NULL if the data cannot be retrieved or no territorial office matches to.

catr_atom_get_address_db_all() returns the following columns:

- territorial_office: Territorial office, corresponding to each province of Spain except the Basque Country and Navarre.
- url: ATOM URL for the corresponding territorial office.
- munic: Name of the municipality.
- date: Reference date of the data. The information from this service is updated twice a year.

catr_atom_get_address_db_to() returns munic, url and date for the selected territorial office.

Source

https://www.catastro.hacienda.gob.es/INSPIRE/buildings/ES.SDGC.BU.atom.xml

See Also

Work with cadastral buildings: catr_atom_get_buildings(), catr_wfs_get_buildings_bbox()

Query ATOM INSPIRE services: catr_atom_get_address(), catr_atom_get_address_db_all(), catr_atom_get_buildings(), catr_atom_get_parcels(), catr_atom_get_parcels_db_all(), catr_atom_search_munic()

Examples

```
catr_atom_get_buildings_db_all()
```

`catr_atom_get_parcels` *ATOM INSPIRE: Download all cadastral parcels for a municipality*

Description

Retrieve spatial data for all cadastral parcels in a municipality using the ATOM INSPIRE service.

Usage

```
catr_atom_get_parcels(
  munic,
  to = NULL,
  what = c("parcel", "zoning"),
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

`munic`	Municipality name, partial name or cadastral code. Use `catr_atom_search_munic()` to find cadastral codes.
`to`	Optional territorial office containing `munic`. Use this argument to narrow the search.
`what`	Information to load, either `"parcel"` for cadastral parcels or `"zoning"` for cadastral zoning.
`cache`	**[Deprecated]** This argument is no longer supported because results are always cached.
`update_cache`	Logical. Whether to refresh the cached file. Defaults to `FALSE`.
`cache_dir`	Path to a cache directory. If `NULL` or `FALSE`, the function stores cached files in a temporary directory. See `base::tempdir()`.
`verbose`	Logical. Whether to display informational messages.

Value

An `sf` object. Returns `NULL` if the data cannot be retrieved.

References

API documentation.

INSPIRE services for cadastral cartography.

See Also

Work with cadastral parcels: catr_atom_get_parcels_db_all(), catr_wfs_get_parcels_bbox()

Query ATOM INSPIRE services: catr_atom_get_address(), catr_atom_get_address_db_all(), catr_atom_get_buildings(), catr_atom_get_buildings_db_all(), catr_atom_get_parcels_db_all(), catr_atom_search_munic()

Examples

```
s <- catr_atom_get_parcels("Melque", to = "Segovia", what = "parcel")

library(ggplot2)

ggplot(s) +
  geom_sf() +
  labs(
    title = "Cadastral parcels",
    subtitle = "Melque de Cercos, Segovia"
  )
```

---

catr_atom_get_parcels_db_all

ATOM INSPIRE: List cadastral parcel download URLs

Description

Create a table of URLs provided by the Spanish Cadastre ATOM INSPIRE service for downloading cadastral parcels.

catr_atom_get_parcels_db_all() provides a summary table with all territorial offices and their municipalities, excluding the Basque Country and Navarre. catr_atom_get_parcels_db_to() provides a table for one territorial office and its municipalities.

Usage

```
catr_atom_get_parcels_db_all(
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)

catr_atom_get_parcels_db_to(
  to,
  cache = deprecated(),
  update_cache = FALSE,
```

```
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

cache	**[Deprecated]** This argument is no longer supported because results are always cached.
update_cache	Logical. Whether to refresh the cached file. Defaults to FALSE.
cache_dir	Path to a cache directory. If NULL or FALSE, the function stores cached files in a temporary directory. See base::tempdir().
verbose	Logical. Whether to display informational messages.
to	Character string. Territorial office to match using base::grep().

Value

A tibble with the requested information. Returns NULL if the data cannot be retrieved or no territorial office matches to.

catr_atom_get_address_db_all() returns the following columns:

- territorial_office: Territorial office, corresponding to each province of Spain except the Basque Country and Navarre.
- url: ATOM URL for the corresponding territorial office.
- munic: Name of the municipality.
- date: Reference date of the data. The information from this service is updated twice a year.

catr_atom_get_address_db_to() returns munic, url and date for the selected territorial office.

Source

https://www.catastro.hacienda.gob.es/INSPIRE/CadastralParcels/ES.SDGC.CP.atom.xml

See Also

Work with cadastral parcels: catr_atom_get_parcels(), catr_wfs_get_parcels_bbox()

Query ATOM INSPIRE services: catr_atom_get_address(), catr_atom_get_address_db_all(), catr_atom_get_buildings(), catr_atom_get_buildings_db_all(), catr_atom_get_parcels(), catr_atom_search_munic()

Examples

```
catr_atom_get_parcels_db_all()
```

`catr_atom_search_munic` *ATOM INSPIRE: Search for municipality codes*

Description

Search for a municipality by name or code and return matching Spanish Cadastre municipality codes.

Usage

```
catr_atom_search_munic(
  munic,
  to = NULL,
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

`munic`	Municipality name, partial name or cadastral code. Use `catr_atom_search_munic()` to find cadastral codes.
`to`	Optional territorial office containing `munic`. Use this argument to narrow the search.
`cache`	**[Deprecated]** This argument is no longer supported because results are always cached.
`update_cache`	Logical. Whether to refresh the cached file. Defaults to `FALSE`.
`cache_dir`	Path to a cache directory. If `NULL` or `FALSE`, the function stores cached files in a temporary directory. See `base::tempdir()`.
`verbose`	Logical. Whether to display informational messages.

Value

A tibble with the territorial office, municipality name and cadastral code. Returns `NULL` if the data cannot be retrieved or no match is found.

See Also

`catr_atom_get_address()`, `catr_atom_get_buildings()` and `catr_atom_get_parcels()` accept the returned cadastral municipality code as `munic`. `catr_get_code_from_coords()` finds that code from coordinates.

Search for cadastral identifiers: `catr_get_code_from_coords()`, `catr_ovc_get_cod_munic()`, `catr_ovc_get_cod_provinces()`

Query ATOM INSPIRE services: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`, `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`, `catr_atom_get_parcels()`, `catr_atom_get_parcels_db_all()`

Examples

```
catr_atom_search_munic("Mad")
```

`catr_clear_cache` *Clear your* R*hrefhttps://CRAN.R-project.org/package=CatastRo***CatastRo** *cache directory*

Description

Use this function with caution. Depending on its arguments, this function:

- Deletes the **CatastRo** configuration directory when `config = TRUE` (`tools::R_user_dir("CatastRo", "config")`).
- Deletes the `cache_dir` directory and its contents when `cached_data = TRUE`.
- Always clears the `CATASTROESP_CACHE_DIR` environment variable.

Usage

```
catr_clear_cache(config = FALSE, cached_data = TRUE, verbose = FALSE)
```

Arguments

`config`	If `TRUE`, deletes the configuration directory of **CatastRo**.
`cached_data`	If `TRUE`, deletes your `cache_dir` and all its contents.
`verbose`	Logical. Whether to display informational messages.

Details

With `config = TRUE` and `cached_data = TRUE`, this function resets the cache state as if you had never used **CatastRo**.

Value

`NULL`, invisibly. This function is called for its side effects.

See Also

`tools::R_user_dir()` defines platform-specific user directories.

Manage the local cache: `catr_set_cache_dir()`

Examples

```
# Don't run this! It modifies your current state
## Not run:
my_cache <- catr_detect_cache_dir()

# Set an example cache
ex <- file.path(tempdir(), "example", "cache")
catr_set_cache_dir(ex, verbose = FALSE)

# Restore initial cache
catr_clear_cache(verbose = TRUE)

catr_set_cache_dir(my_cache)
identical(my_cache, catr_detect_cache_dir())

## End(Not run)
```

`catr_get_code_from_coords` *Get a cadastral municipality code from coordinates*

Description

Retrieve the municipality code associated with an `sf` object or a coordinate pair.

Usage

```
catr_get_code_from_coords(
  x,
  srs = NULL,
  verbose = FALSE,
  cache_dir = NULL,
  ...
)
```

Arguments

`x`	Coordinate input. It can be: • A pair of coordinates `c(x, y)`. In this case the `srs` of the coordinates must be provided. • An `sf` object. If the object has several geometries, only the first geometry is used. This function extracts coordinates using `sf::st_centroid(x, of_largest_polygon = TRUE)`.
`srs`	SRS/CRS to use in the query. To see allowed values, use catr_srs_values, specifically the `ovc_service` column.
`verbose`	Logical. Whether to display informational messages.

cache_dir Path to a cache directory. If NULL or FALSE, the function stores cached files in a temporary directory. See base::tempdir().

... Arguments passed on to mapSpain::esp_get_munic_siane

year Character string or number. Release year. It must use format YYYY (assuming end of year) or YYYY-MM-DD. Historical information starts as of 2005.

resolution Character string or number. Resolution of the geospatial data. One of:

- "10": 1:10 million.
- "6.5": 1:6.5 million.
- "3": 1:3 million.

region Optional. A vector of region names, NUTS or ISO codes (see esp_dict_region_code()).

munic Character string. A name or regex expression with the names of the required municipalities. Use NULL to return all municipalities.

Details

On a successful query, this function returns a one-row tibble with the following columns:

- munic: Municipality name used by the Spanish Cadastre.
- catr_to: Cadastral territorial office code.
- catr_munic: Municipality code as recorded by the Cadastre.
- catrcode: Full cadastral code for the municipality.
- cpro: Province code according to the INE.
- cmun: Municipality code according to the INE.
- inecode: Full INE code for the municipality.
- Remaining fields: See the API documentation.

Value

A tibble as described in **Details**. Returns NULL if the request fails.

See Also

- mapSpain::esp_get_munic_siane() retrieves municipality geometries.
- catr_ovc_get_cod_munic() retrieves municipality codes.
- sf::st_centroid() computes geometry centroids.

Search for cadastral identifiers: catr_atom_search_munic(), catr_ovc_get_cod_munic(), catr_ovc_get_cod_provinc

Examples

```
# Use with coordinates
catr_get_code_from_coords(c(-16.25462, 28.46824), srs = 4326)

# Use with an `sf` object.
prov <- mapSpain::esp_get_prov("Caceres")
```

```
catr_get_code_from_coords(prov)
```

`catr_ovc_get_cod_munic` *OVCCallejero: Get municipality codes*

Description

Query the OVCCallejero ConsultaMunicipioCodigos service to retrieve municipality names and codes from the Spanish Cadastre and the National Statistics Institute (INE).

Usage

```
catr_ovc_get_cod_munic(cpro, cmun = NULL, cmun_ine = NULL, verbose = FALSE)
```

Arguments

`cpro`	Province code returned by `catr_ovc_get_cod_provinces()`.
`cmun`, `cmun_ine`	Municipality code as recorded by the Spanish Cadastre (`cmun`) or the National Statistics Institute (`cmun_ine`). Either `cmun` or `cmun_ine` must be provided.
`verbose`	Logical. Whether to display informational messages.

Details

On a successful query, this function returns a one-row tibble with the following columns:

- `munic`: Municipality name used by the Spanish Cadastre.
- `catr_to`: Cadastral territorial office code.
- `catr_munic`: Municipality code as recorded by the Cadastre.
- `catrcode`: Full cadastral code for the municipality.
- `cpro`: Province code according to the INE.
- `cmun`: Municipality code according to the INE.
- `inecode`: Full INE code for the municipality.
- Remaining fields: See the API documentation.

Value

A tibble as described in **Details**. Returns `NULL` if the request fails.

References

ConsultaMunicipioCodigos.

See Also

catr_atom_search_munic() searches municipality codes by name. catr_atom_get_parcels() accepts the returned catrcode as munic. mapSpain::esp_get_munic_siane() retrieves municipality geometries, including the INE code.

Search for cadastral identifiers: catr_atom_search_munic(), catr_get_code_from_coords(), catr_ovc_get_cod_provinces()

Query OVC web services: catr_ovc_get_cod_provinces(), catr_ovc_get_cpmrc(), catr_ovc_get_rccoor(), catr_ovc_get_rccoor_distancia()

Examples

```
# Get municipality by cadastral code
ab <- catr_ovc_get_cod_munic(cpro = 2, cmun = 900)

ab

# Same query using the INE code

ab2 <- catr_ovc_get_cod_munic(cpro = 2, cmun_ine = 3)

ab2
```

---

catr_ovc_get_cod_provinces

OVCCallejero: Get province codes

Description

Query the OVCCallejero ConsultaProvincia service to retrieve provinces and their INE codes. Only provinces within the jurisdiction of the Directorate General for Cadastre are included.

Usage

```
catr_ovc_get_cod_provinces(verbose = FALSE)
```

Arguments

verbose	Logical. Whether to display informational messages.

Value

A tibble with province names and codes. Returns NULL if the request fails.

References

ConsultaProvincia.

See Also

`catr_ovc_get_cod_munic()` uses a province code as `cpro` to look up municipality codes.

Search for cadastral identifiers: `catr_atom_search_munic()`, `catr_get_code_from_coords()`, `catr_ovc_get_cod_munic()`

Query OVC web services: `catr_ovc_get_cod_munic()`, `catr_ovc_get_cpmrc()`, `catr_ovc_get_rccoor()`, `catr_ovc_get_rccoor_distancia()`

Examples

```
catr_ovc_get_cod_provinces()
```

---

catr_ovc_get_cpmrc *OVCCoordenadas: Geocode a cadastral reference*

Description

Query the OVCCoordenadas Consulta CPMRC service to retrieve coordinates for a parcel reference. The returned coordinates locate the parcel centroid.

Usage

```
catr_ovc_get_cpmrc(
  rc,
  srs = 4326,
  province = NULL,
  municipality = NULL,
  verbose = FALSE
)
```

Arguments

`rc`	A 14-character cadastral parcel reference to geocode.
`srs`	SRS/CRS to use in the query. To see allowed values, use catr_srs_values, specifically the `ovc_service` column.
`province, municipality`	Optional character strings used to narrow the search. `province` is required when `municipality` is provided.
`verbose`	Logical. Whether to display informational messages.

Details

If the API returns no results or reports an error, the result is a tibble containing only query information.

On a successful query, this function returns a tibble with one row per cadastral reference, including the following columns:

- `xcoord`, `ycoord`: X and Y coordinates in the specified SRS.
- `refcat`: Cadastral reference.
- `address`: Address as recorded in the Spanish Cadastre.
- Remaining fields: See the API documentation.

Value

A tibble as described in **Details**. Returns `NULL` if the request fails.

References

Consulta CPMRC.

See Also

- catr_srs_values lists supported SRS values.
- `vignette("ovcservice", package = "CatastRo")` describes the OVC services.

Work with cadastral references: `catr_ovc_get_rccoor()`, `catr_ovc_get_rccoor_distancia()`

Query OVC web services: `catr_ovc_get_cod_munic()`, `catr_ovc_get_cod_provinces()`, `catr_ovc_get_rccoor()`, `catr_ovc_get_rccoor_distancia()`

Examples

```
# Using all arguments
catr_ovc_get_cpmrc("13077A01800039",
  4230,
  province = "CIUDAD REAL",
  municipality = "SANTA CRUZ DE MUDELA"
)

# Only the cadastral reference
catr_ovc_get_cpmrc("9872023VH5797S")
```

`catr_ovc_get_rccoor` *OVCCoordenadas: Reverse geocode coordinates*

Description

Query the OVCCoordenadas Consulta RCCOOR service to retrieve the cadastral reference associated with a pair of coordinates.

Usage

```
catr_ovc_get_rccoor(lat, lon, srs = 4326, verbose = FALSE)
```

Arguments

`lat`	Y coordinate for the query, expressed in the SRS/CRS defined by `srs`. For geographic coordinates, this is the latitude.
`lon`	X coordinate for the query, expressed in the SRS/CRS defined by `srs`. For geographic coordinates, this is the longitude.
`srs`	SRS/CRS to use in the query. To see allowed values, use catr_srs_values, specifically the `ovc_service` column.
`verbose`	Logical. Whether to display informational messages.

Details

If the API returns no results or reports an error, the result is a tibble containing only query information.

On a successful query, this function returns a tibble with one row per cadastral reference, including the following columns:

- `geo.xcen`, `geo.ycen`, `geo.srs`: Input arguments of the query.
- `refcat`: Cadastral reference.
- `address`: Address as recorded in the Spanish Cadastre.
- Remaining fields: See the API documentation.

Value

A tibble as described in **Details**. Returns `NULL` if the request fails.

References

Consulta RCCOOR.

See Also

`catr_ovc_get_rccoor_distancia()` also searches a nearby area when no exact match is found. `catr_wfs_get_parcels_parcel()` retrieves parcel geometries using the returned cadastral references.

Work with cadastral references: `catr_ovc_get_cpmrc()`, `catr_ovc_get_rccoor_distancia()`

Query OVC web services: `catr_ovc_get_cod_munic()`, `catr_ovc_get_cod_provinces()`, `catr_ovc_get_cpmrc()`, `catr_ovc_get_rccoor_distancia()`

Examples

```
catr_ovc_get_rccoor(
  lat = 38.6196566583596,
  lon = -3.45624183836806,
  srs = 4326
)
```

---

catr_ovc_get_rccoor_distancia

OVCCoordenadas: Find cadastral references near coordinates

Description

Query the OVCCoordenadas Consulta RCCOOR Distancia service to retrieve cadastral references near a pair of coordinates. If no exact match is found, the API searches a square with sides of 50 meters, centered on the requested coordinates.

Usage

```
catr_ovc_get_rccoor_distancia(lat, lon, srs = 4326, verbose = FALSE)
```

Arguments

`lat`	Y coordinate for the query, expressed in the SRS/CRS defined by `srs`. For geographic coordinates, this is the latitude.
`lon`	X coordinate for the query, expressed in the SRS/CRS defined by `srs`. For geographic coordinates, this is the longitude.
`srs`	SRS/CRS to use in the query. To see allowed values, use catr_srs_values, specifically the `ovc_service` column.
`verbose`	Logical. Whether to display informational messages.

Details

If the API returns no results or reports an error, the result is a tibble containing only query information.

On a successful query, this function returns a tibble with one row per cadastral reference, including the following columns:

- `geo.xcen`, `geo.ycen`, `geo.srs`: Input arguments of the query.
- `refcat`: Cadastral reference.
- `address`: Address as recorded in the Spanish Cadastre.
- `cmun_ine`: Full five-digit INE municipality code, combining the province and municipality codes (National Statistics Institute).
- `dis`: Distance from the cadastral reference to the queried point.
- Remaining fields: See the API documentation.

Value

A tibble as described in **Details**. Returns `NULL` if the request fails.

References

Consulta RCCOOR Distancia.

See Also

`catr_ovc_get_rccoor()` looks up the cadastral reference at the exact coordinates. `catr_wfs_get_parcels_parcel()` retrieves parcel geometries using the returned cadastral references.

Work with cadastral references: `catr_ovc_get_cpmrc()`, `catr_ovc_get_rccoor()`

Query OVC web services: `catr_ovc_get_cod_munic()`, `catr_ovc_get_cod_provinces()`, `catr_ovc_get_cpmrc()`, `catr_ovc_get_rccoor()`

Examples

```
catr_ovc_get_rccoor_distancia(
  lat = 40.963200,
  lon = -5.671420,
  srs = 4326
)
```

catr_set_cache_dir *Set your* R*hrefhttps://CRAN.R-project.org/package=CatastRo***CatastRo** *cache directory*

Description

Configure the cache directory used by **CatastRo**. Use `Sys.getenv("CATASTROESP_CACHE_DIR")` or `catr_detect_cache_dir()` to inspect the current path.

Usage

```
catr_set_cache_dir(
  cache_dir = NULL,
  overwrite = FALSE,
  install = FALSE,
  verbose = TRUE
)

catr_detect_cache_dir()
```

Arguments

`cache_dir`	Path to a cache directory. If `NULL` or `FALSE`, the function stores cached files in a temporary directory. See `base::tempdir()`.
`overwrite`	Logical. Whether to overwrite an existing `CATASTROESP_CACHE_DIR` value already present on your machine.
`install`	Logical. Whether to store the path locally for use in future sessions. Defaults to `FALSE`.
`verbose`	Logical. Whether to display informational messages.

Details

By default, when no `cache_dir` is set, **CatastRo** uses a directory inside `base::tempdir()`, so files are temporary and are removed when the R session ends. To persist a cache across R sessions, use `catr_set_cache_dir(cache_dir, install = TRUE)`, which writes the chosen path to a small configuration file under `tools::R_user_dir("CatastRo", "config")`.

Value

`catr_set_cache_dir()` returns a character string containing the cache directory path, invisibly. This function is called for its side effects.

`catr_detect_cache_dir()` returns a character string containing the cache directory path used in this session.

Caching strategies

Source files are cached after download. **CatastRo** implements the following caching options:

- For occasional use, rely on the default `tempdir()`-based cache without installing it.
- Modify the cache for a single session by setting `catr_set_cache_dir(cache_dir = "a/path/here")`.
- For reproducible workflows, install a persistent cache with `catr_set_cache_dir(cache_dir = "a/path/here", install = TRUE)`. This cache is kept across R sessions.
- For caching specific files, use the `cache_dir` argument in the corresponding function.

Cached files can occasionally become corrupt. In that case, try downloading the data by setting `update_cache = TRUE` in the corresponding function.

If a download fails, try another download method and save the file in `cache_dir`. Use `verbose = TRUE` to inspect the API query and `catr_detect_cache_dir()` to identify your cache path.

Note

In **CatastRo** >= 1.0.0, the location of the configuration file has moved from `rappdirs::user_config_dir("CatastRo", "R")` to `tools::R_user_dir("CatastRo", "config")`. A migration function moves previous configuration files to the new location and displays a message. This message appears only once.

See Also

`tools::R_user_dir()` defines platform-specific user directories.

Manage the local cache: `catr_clear_cache()`

Examples

```
# Caution! This modifies your current state
## Not run:
my_cache <- catr_detect_cache_dir()

# Set an example cache
ex <- file.path(tempdir(), "example", "cachenew")
catr_set_cache_dir(ex)

catr_detect_cache_dir()

# Restore initial cache
catr_set_cache_dir(my_cache)
identical(my_cache, catr_detect_cache_dir())

## End(Not run)

catr_detect_cache_dir()
```

catr_srs_values *Reference SRS codes for* R*hrefhttps://CRAN.R-project.org/package=CatastRo***CatastRo** *services*

Description

A tibble containing valid SRS values, also known as CRS values, for each API service. Values are represented as EPSG codes.

Format

A tibble with 16 rows and columns:

SRS Spatial reference system (SRS) value, also known as a coordinate reference system (CRS), identified by the corresponding EPSG code.

Description Description of the SRS/EPSG code.

ovc_service Logical. Whether this code is valid for OVC services.

wfs_service Logical. Whether this code is valid for WFS INSPIRE services.

Details

Table: Content of catr_srs_values

SRS	Description	ovc_service	wfs_service
3785	Web Mercator	FALSE	TRUE
3857	Web Mercator	FALSE	TRUE
4230	Geográficas en ED 50	TRUE	FALSE
4258	Geográficas en ETRS89	TRUE	TRUE
4326	Geográficas en WGS 84	TRUE	TRUE
23029	UTM huso 29N en ED50	TRUE	FALSE
23030	UTM huso 30N en ED50	TRUE	FALSE
23031	UTM huso 31N en ED50	TRUE	FALSE
25829	UTM huso 29N en ETRS89	TRUE	TRUE
25830	UTM huso 30N en ETRS89	TRUE	TRUE
25831	UTM huso 31N en ETRS89	TRUE	TRUE
32627	UTM huso 27N en WGS 84	TRUE	FALSE
32628	UTM huso 28N en WGS 84	TRUE	FALSE
32629	UTM huso 29N en WGS 84	TRUE	FALSE
32630	UTM huso 30N en WGS 84	TRUE	FALSE
32631	UTM huso 31N en WGS 84	TRUE	FALSE

References

- OVCCoordenadas.
- WFS INSPIRE service.

See Also

`catr_ovc_get_cpmrc()` and `catr_ovc_get_rccoor()` accept SRS codes listed in the `ovc_service` column. `catr_wfs_get_parcels_bbox()` accepts codes listed in the `wfs_service` column. `sf::st_crs()` inspects the corresponding coordinate reference system.

Examples

```
data("catr_srs_values")

# OVC valid codes
library(dplyr)

catr_srs_values |> filter(ovc_service)

# WFS valid codes

catr_srs_values |> filter(wfs_service)

# Use with sf::st_crs()

catr_srs_values |>
  filter(wfs_service & ovc_service) |>
  print() |>
  # Select the first value.
  slice_head(n = 1) |>
  pull(SRS) |>
  # Convert to a CRS.
  sf::st_crs(.)
```

`catr_wfs_get_address_bbox`

WFS INSPIRE: Download addresses

Description

Retrieve spatial address data through several types of WFS queries:

- By bounding box: `catr_wfs_get_address_bbox()` retrieves objects included in the provided bounding box. See **Bounding box**.
- By street code: `catr_wfs_get_address_codvia()` retrieves objects for specific addresses.
- By cadastral reference: `catr_wfs_get_address_rc()` retrieves objects for specific cadastral references.
- By postal codes: `catr_wfs_get_address_postalcode()` retrieves objects for specific postal codes.

Usage

```
catr_wfs_get_address_bbox(x, srs = NULL, verbose = FALSE)

catr_wfs_get_address_codvia(codvia, del, mun, srs = NULL, verbose = FALSE)

catr_wfs_get_address_rc(rc, srs = NULL, verbose = FALSE)

catr_wfs_get_address_postalcode(postalcode, srs = NULL, verbose = FALSE)
```

Arguments

x	Input defining the query area. See **Bounding box**. It can be: • A numeric vector of length 4 with the coordinates that define the bounding box: `c(xmin, ymin, xmax, ymax)`. • An `sf` or `sfc` object from **sf**.
srs	SRS/CRS to use in the query. To see allowed values, use catr_srs_values, specifically the `wfs_service` column. See **Bounding box**.
verbose	Logical. Whether to display informational messages.
codvia	Cadastral street code.
del	Cadastral office code.
mun	Cadastral municipality code.
rc	Cadastral reference to retrieve.
postalcode	Postal code.

Value

An `sf` object. Returns `NULL` if the data cannot be retrieved.

API limits

The API service is limited to a bounding box of 4 km2 and a maximum of 5,000 elements.

Bounding box

When `x` is a numeric vector, make sure that `srs` matches the coordinate values. This function queries the bounding box in EPSG:25830, ETRS89 / UTM zone 30N, to work around a potential API issue.

When `x` is an `sf` object, the `srs` value is ignored. In this case, the bounding box of the `sf` object is used for the query (see `sf::st_bbox()`).

The result uses the SRS of the input spatial object or the `srs` value provided with a numeric vector.

References

API documentation.

INSPIRE services for cadastral cartography.

See Also

`catr_wms_get_layer()` downloads a map image using the returned spatial object as its extent (`x`).

Work with cadastral addresses: `catr_atom_get_address()`, `catr_atom_get_address_db_all()`

Query WFS INSPIRE services: `catr_wfs_get_buildings_bbox()`, `catr_wfs_get_parcels_bbox()`, `inspire_wfs_get()`

Examples

```
ad <- catr_wfs_get_address_bbox(
  c(
    233673, 4015968, 233761, 4016008
  ),
  srs = 25830
)

library(ggplot2)

ggplot(ad) +
  geom_sf()
```

---

catr_wfs_get_buildings_bbox

WFS INSPIRE: Download buildings

Description

Retrieve spatial building data through two types of WFS queries:

- By bounding box: `catr_wfs_get_buildings_bbox()` retrieves objects included in the provided bounding box. See **Bounding box**.
- By cadastral reference: `catr_wfs_get_buildings_rc()` retrieves objects for specific cadastral references.

Usage

```
catr_wfs_get_buildings_bbox(
  x,
  what = c("building", "buildingpart", "other"),
  srs = NULL,
  verbose = FALSE
)

catr_wfs_get_buildings_rc(
  rc,
```

```
  what = c("building", "buildingpart", "other"),
  srs = NULL,
  verbose = FALSE
)
```

Arguments

x	Input defining the query area. See **Bounding box**. It can be: • A numeric vector of length 4 with the coordinates that define the bounding box: `c(xmin, ymin, xmax, ymax)`. • An `sf` or `sfc` object from **sf**.
what	Information to load, either `"building"` for buildings, `"buildingpart"` for parts of a building or `"other"` for other elements such as swimming pools.
srs	SRS/CRS to use in the query. To see allowed values, use catr_srs_values, specifically the `wfs_service` column. See **Bounding box**.
verbose	Logical. Whether to display informational messages.
rc	Cadastral reference to retrieve.

Value

An `sf` object. Returns `NULL` if the data cannot be retrieved.

API limits

The API service is limited to a bounding box of 4 km2 and a maximum of 5,000 elements.

Bounding box

When `x` is a numeric vector, make sure that `srs` matches the coordinate values. This function queries the bounding box in EPSG:25830, ETRS89 / UTM zone 30N, to work around a potential API issue.

When `x` is an `sf` object, the `srs` value is ignored. In this case, the bounding box of the `sf` object is used for the query (see `sf::st_bbox()`).

The result uses the SRS of the input spatial object or the `srs` value provided with a numeric vector.

References

API documentation.

INSPIRE services for cadastral cartography.

See Also

`catr_wms_get_layer()` downloads a map image using the returned spatial object as its extent (`x`).

Work with cadastral buildings: `catr_atom_get_buildings()`, `catr_atom_get_buildings_db_all()`

Query WFS INSPIRE services: `catr_wfs_get_address_bbox()`, `catr_wfs_get_parcels_bbox()`, `inspire_wfs_get()`

Examples

```
# Using a bounding box
building <- catr_wfs_get_buildings_bbox(
  c(
    376550,
    4545424,
    376600,
    4545474
  ),
  srs = 25830
)
library(ggplot2)
ggplot(building) +
  geom_sf() +
  labs(title = "Search using a bounding box")

# Using a cadastral reference
rc <- catr_wfs_get_buildings_rc("6656601UL7465N")
library(ggplot2)
ggplot(rc) +
  geom_sf() +
  labs(title = "Search using rc")
```

`catr_wfs_get_parcels_bbox` *WFS INSPIRE: Download cadastral parcels*

Description

Retrieve spatial cadastral parcel and zoning data through several types of WFS queries:

- By bounding box: `catr_wfs_get_parcels_bbox()` retrieves objects included in the provided bounding box. See **Bounding box**.
- By zoning: `catr_wfs_get_parcels_zoning()` retrieves objects for a specific cadastral zone.
- By cadastral parcel: `catr_wfs_get_parcels_parcel()` retrieves cadastral parcels for a specific cadastral reference.
- Neighbor cadastral parcels: `catr_wfs_get_parcels_neigh_parcel()` retrieves neighboring cadastral parcels for a specific cadastral reference.
- Cadastral parcels by zoning: `catr_wfs_get_parcels_parcel_zoning()` retrieves cadastral parcels for a specific cadastral zone.

Usage

```
catr_wfs_get_parcels_bbox(
  x,
  what = c("parcel", "zoning"),
  srs = NULL,
  verbose = FALSE
)

catr_wfs_get_parcels_zoning(cod_zona, srs = NULL, verbose = FALSE)

catr_wfs_get_parcels_parcel(rc, srs = NULL, verbose = FALSE)

catr_wfs_get_parcels_neigh_parcel(rc, srs = NULL, verbose = FALSE)

catr_wfs_get_parcels_parcel_zoning(cod_zona, srs = NULL, verbose = FALSE)
```

Arguments

x	Input defining the query area. See **Bounding box**. It can be: • A numeric vector of length 4 with the coordinates that define the bounding box: `c(xmin, ymin, xmax, ymax)`. • An `sf` or `sfc` object from **sf**.
what	Information to load, either `"parcel"` for cadastral parcels or `"zoning"` for cadastral zoning.
srs	SRS/CRS to use in the query. To see allowed values, use catr_srs_values, specifically the `wfs_service` column. See **Bounding box**.
verbose	Logical. Whether to display informational messages.
cod_zona	Cadastral zone code.
rc	Cadastral reference to retrieve.

Value

An `sf` object. Returns `NULL` if the data cannot be retrieved.

API limits

The API service has the following limits:

- `"parcel"`: Bounding box of 1 km2 and a maximum of 5,000 elements.
- `"zoning"`: Bounding box of 25 km2 and a maximum of 5,000 elements.

Bounding box

When `x` is a numeric vector, make sure that `srs` matches the coordinate values. This function queries the bounding box in EPSG:25830, ETRS89 / UTM zone 30N, to work around a potential API issue.

When x is an sf object, the srs value is ignored. In this case, the bounding box of the sf object is used for the query (see sf::st_bbox()).

The result uses the SRS of the input spatial object or the srs value provided with a numeric vector.

References

API documentation.

INSPIRE services for cadastral cartography.

See Also

catr_wms_get_layer() downloads a map image using the returned spatial object as its extent (x).

Work with cadastral parcels: catr_atom_get_parcels(), catr_atom_get_parcels_db_all()

Query WFS INSPIRE services: catr_wfs_get_address_bbox(), catr_wfs_get_buildings_bbox(), inspire_wfs_get()

Examples

```
cp <- catr_wfs_get_parcels_bbox(
  c(
    233673, 4015968, 233761, 4016008
  ),
  srs = 25830
)

library(ggplot2)

ggplot(cp) +
  geom_sf()
```

---

catr_wms_get_layer *WMS INSPIRE: Download georeferenced map images*

Description

Retrieve georeferenced map images from the Spanish Cadastre WMS service. This function wraps mapSpain::esp_get_tiles().

Usage

```
catr_wms_get_layer(
  x,
  srs = NULL,
 what = c("building", "buildingpart", "parcel", "zoning", "address", "admboundary",
```

```
    "admunit"),
  styles = "default",
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  crop = FALSE,
  options = NULL,
  ...
)
```

Arguments

x	Input defining the query area. See **Bounding box**. It can be: • A numeric vector of length 4 with the coordinates that define the bounding box: `c(xmin, ymin, xmax, ymax)`. • An `sf` or `sfc` object from **sf**.
srs	SRS/CRS to use in the query. To see allowed values, use catr_srs_values, specifically the `wfs_service` column. See **Bounding box**.
what	WMS layer to download. See **Layers and styles**.
styles	Style to apply to the selected WMS layer. See **Layers and styles**.
update_cache	Logical. Whether to refresh the cached file. Defaults to `FALSE`.
cache_dir	Path to a cache directory. If `NULL` or `FALSE`, the function stores cached files in a temporary directory. See `base::tempdir()`.
verbose	Logical. Whether to display informational messages.
crop	Logical. If `TRUE`, crop results to the specified `x` extent. If `x` is an `sf` object with one `POINT`, `crop` is set to `FALSE`. See `terra::crop()`.
options	A named list containing additional options to pass to the query.
...	Arguments passed on to `mapSpain::esp_get_tiles` `res` Character string or number. Only valid for WMS providers. Resolution (in pixels) of the final tile. `bbox_expand` Number. Expansion percentage of the bounding box of `x`. `transparent` Logical. Whether to use a transparent background, if supported. `mask` Logical. `TRUE` to mask the result to `x`. See `terra::mask()`.

Value

A `SpatRaster` with three RGB or four RGBA layers. See `terra::RGB()`.

Bounding box

When `x` is a numeric vector, make sure that `srs` matches the coordinate values. When `x` is an `sf` object, the `srs` value is ignored.

The query uses EPSG:3857 (Web Mercator), then transforms the tile back to the SRS of `x`. If the tile appears distorted, provide a spatial object as `x` or set `srs` to the SRS of the requested tile. See **Examples**.

Layers and styles

Layers:

The what argument selects one of the following API layers:

- "parcel": CP.CadastralParcel.
- "zoning": CP.CadastralZoning.
- "building": BU.Building.
- "buildingpart": BU.BuildingPart.
- "address": AD.Address.
- "admboundary": AU.AdministrativeBoundary.
- "admunit": AU.AdministrativeUnit.

Styles:

The WMS service provides different styles for each layer (what argument). Available styles include:

- "parcel": "BoundariesOnly", "ReferencePointOnly" and "ELFCadastre".
- "zoning": "BoundariesOnly" and "ELFCadastre".
- "building" and "buildingpart": "ELFCadastre".
- "address": "Number.ELFCadastre".
- "admboundary" and "admunit": "ELFCadastre".

See the API documentation for complete layer and style information.

See Also

- catr_wfs_get_address_bbox(), catr_wfs_get_buildings_bbox() and catr_wfs_get_parcels_bbox() retrieve vector geometries that can define the map extent through x.
- mapSpain::esp_get_tiles() downloads map tiles.
- terra::RGB() identifies RGB channels.
- terra::plotRGB() and tidyterra::geom_spatraster_rgb() plot RGB rasters.

Examples

```
if (requireNamespace("tidyterra", quietly = TRUE)) {
  # With a bounding box

  pict <- catr_wms_get_layer(
    c(222500, 4019500, 223700, 4020700),
    srs = 25830,
    what = "parcel"
  )

  library(mapSpain)
  library(ggplot2)
  library(tidyterra)

  ggplot() +
```

```
    geom_spatraster_rgb(data = pict)

  # With a spatial object

  parcels <- catr_wfs_get_parcels_neigh_parcel("3662303TF3136B", srs = 25830)

  # Use styles

  parcels_img <- catr_wms_get_layer(parcels,
    what = "buildingpart",
    srs = 25830, # Same as the parcels object
    bbox_expand = 0.3,
    styles = "ELFCadastre"
  )

  ggplot() +
    geom_sf(data = parcels, fill = "blue", alpha = 0.5) +
    geom_spatraster_rgb(data = parcels_img)
}
```

`inspire_wfs_get` *Query WFS INSPIRE services*

Description

Build and run a WFS INSPIRE request. This function supports the package's WFS functions and is also available for querying other cadastral or INSPIRE resources.

Usage

```
inspire_wfs_get(
  scheme = "https",
  hostname = "ovc.catastro.meh.es",
  path = "INSPIRE/wfsCP.aspx",
  query = list(),
  cache_dir = NULL,
  verbose = FALSE
)
```

Arguments

`scheme`	Character string specifying the protocol used to access the resource.
`hostname`	Character string specifying the resource host.
`path`	Character string specifying the resource path on the host.
`query`	Named list of query parameters and their values.

cache_dir	Path to a cache directory. If NULL or FALSE, the function stores cached files in a temporary directory. See base::tempdir().
verbose	Logical. Whether to display informational messages.

Details

This function constructs a request URL from its components, downloads the result to the cache directory and reports WFS exceptions. See **Examples**.

Value

A character string containing the downloaded file path. Returns NULL if the request fails.

See Also

Query WFS INSPIRE services: catr_wfs_get_address_bbox(), catr_wfs_get_buildings_bbox(), catr_wfs_get_parcels_bbox()

Examples

```
# Access the Cadastre of Navarre.
# See also https://ropenspain.github.io/CatastRoNav/

file_local <- inspire_wfs_get(
  hostname = "inspire.navarra.es",
  path = "services/BU/wfs",
  query = list(
    service = "WFS",
    request = "getfeature",
    typenames = "BU:Building",
    bbox = "609800,4740100,611000,4741300",
    SRSNAME = "EPSG:25830"
  )
)

if (!is.null(file_local)) {
  pamp <- sf::read_sf(file_local)

  if (requireNamespace("ggplot2", quietly = TRUE)) {
    library(ggplot2)
    ggplot(pamp) +
      geom_sf()
  }
}
```

Index

∗ **addresses**
  catr_atom_get_address, 2
  catr_atom_get_address_db_all, 4
  catr_wfs_get_address_bbox, 26
∗ **atom_services**
  catr_atom_get_address, 2
  catr_atom_get_address_db_all, 4
  catr_atom_get_buildings, 5
  catr_atom_get_buildings_db_all, 7
  catr_atom_get_parcels, 9
  catr_atom_get_parcels_db_all, 10
  catr_atom_search_munic, 12
∗ **buildings**
  catr_atom_get_buildings, 5
  catr_atom_get_buildings_db_all, 7
  catr_wfs_get_buildings_bbox, 28
∗ **cache_utilities**
  catr_clear_cache, 13
  catr_set_cache_dir, 23
∗ **cadastral_references**
  catr_ovc_get_cpmrc, 18
  catr_ovc_get_rccoor, 20
  catr_ovc_get_rccoor_distancia, 21
∗ **datasets**
  catr_srs_values, 25
∗ **ovc_services**
  catr_ovc_get_cod_munic, 16
  catr_ovc_get_cod_provinces, 17
  catr_ovc_get_cpmrc, 18
  catr_ovc_get_rccoor, 20
  catr_ovc_get_rccoor_distancia, 21
∗ **ovccallejero_services**
  catr_get_code_from_coords, 14
  catr_ovc_get_cod_munic, 16
  catr_ovc_get_cod_provinces, 17
∗ **ovccoordenadas_services**
  catr_ovc_get_cpmrc, 18
  catr_ovc_get_rccoor, 20
  catr_ovc_get_rccoor_distancia, 21
∗ **parcels**
  catr_atom_get_parcels, 9
  catr_atom_get_parcels_db_all, 10
  catr_wfs_get_parcels_bbox, 30
∗ **search_tools**
  catr_atom_search_munic, 12
  catr_get_code_from_coords, 14
  catr_ovc_get_cod_munic, 16
  catr_ovc_get_cod_provinces, 17
∗ **wfs_services**
  catr_wfs_get_address_bbox, 26
  catr_wfs_get_buildings_bbox, 28
  catr_wfs_get_parcels_bbox, 30
  inspire_wfs_get, 35
∗ **wms_services**
  catr_wms_get_layer, 32

base::grep(), *5*, *8*, *11*
base::tempdir(), *3, 4*, *6*, *8, 9*, *11, 12*, *15*, *23*, *33*, *36*

catr_atom_get_address, 2
catr_atom_get_address(), *5, 6*, *8*, *10–13*, *28*
catr_atom_get_address_db_all, 4
catr_atom_get_address_db_all(), *3, 4*, *6*, *8*, *10, 11*, *13*, *28*
catr_atom_get_address_db_to *(*catr_atom_get_address_db_all*)*, 4
catr_atom_get_address_db_to(), *4*
catr_atom_get_buildings, 5
catr_atom_get_buildings(), *3*, *5*, *8*, *10–13*, *29*
catr_atom_get_buildings_db_all, 7
catr_atom_get_buildings_db_all(), *3*, *5–7*, *10, 11*, *13*, *29*
catr_atom_get_buildings_db_to *(*catr_atom_get_buildings_db_all*)*, 7

catr_atom_get_buildings_db_to(), *7*
catr_atom_get_parcels, 9
catr_atom_get_parcels(), *3*, *5, 6*, *8*, *11–13*, *17*, *32*
catr_atom_get_parcels_db_all, 10
catr_atom_get_parcels_db_all(), *3*, *5, 6*, *8*, *10*, *13*, *32*
catr_atom_get_parcels_db_to *(*catr_atom_get_parcels_db_all*)*, 10
catr_atom_get_parcels_db_to(), *10*
catr_atom_search_munic, 12
catr_atom_search_munic(), *3*, *5, 6*, *8–12*, *15*, *17, 18*
catr_clear_cache, 13
catr_clear_cache(), *24*
catr_detect_cache_dir *(*catr_set_cache_dir*)*, 23
catr_detect_cache_dir(), *23, 24*
catr_get_code_from_coords, 14
catr_get_code_from_coords(), *12*, *17, 18*
catr_ovc_get_cod_munic, 16
catr_ovc_get_cod_munic(), *12*, *15*, *18, 19*, *21, 22*
catr_ovc_get_cod_provinces, 17
catr_ovc_get_cod_provinces(), *12*, *15–17*, *19*, *21, 22*
catr_ovc_get_cpmrc, 18
catr_ovc_get_cpmrc(), *17, 18*, *21, 22*, *26*
catr_ovc_get_rccoor, 20
catr_ovc_get_rccoor(), *17–19*, *22*, *26*
catr_ovc_get_rccoor_distancia, 21
catr_ovc_get_rccoor_distancia(), *17–19*, *21*
catr_set_cache_dir, 23
catr_set_cache_dir(), *13*
catr_srs_values, *14*, *18–21*, *25*, 25, *27*, *29*, *31*, *33*
catr_wfs_get_address_bbox, 26
catr_wfs_get_address_bbox(), *3*, *5*, *29*, *32*, *34*, *36*
catr_wfs_get_address_codvia *(*catr_wfs_get_address_bbox*)*, 26
catr_wfs_get_address_postalcode *(*catr_wfs_get_address_bbox*)*, 26
catr_wfs_get_address_rc *(*catr_wfs_get_address_bbox*)*, 26
catr_wfs_get_buildings_bbox, 28
catr_wfs_get_buildings_bbox(), *6*, *8*, *28*, *32*, *34*, *36*
catr_wfs_get_buildings_rc *(*catr_wfs_get_buildings_bbox*)*, 28
catr_wfs_get_parcels_bbox, 30
catr_wfs_get_parcels_bbox(), *10, 11*, *26*, *28, 29*, *34*, *36*
catr_wfs_get_parcels_neigh_parcel *(*catr_wfs_get_parcels_bbox*)*, 30
catr_wfs_get_parcels_parcel *(*catr_wfs_get_parcels_bbox*)*, 30
catr_wfs_get_parcels_parcel(), *21, 22*
catr_wfs_get_parcels_parcel_zoning *(*catr_wfs_get_parcels_bbox*)*, 30
catr_wfs_get_parcels_zoning *(*catr_wfs_get_parcels_bbox*)*, 30
catr_wms_get_layer, 32
catr_wms_get_layer(), *28, 29*, *32*
character, *23*, *36*

esp_dict_region_code(), *15*

inspire_wfs_get, 35
inspire_wfs_get(), *28, 29*, *32*

mapSpain::esp_get_munic_siane, *15*
mapSpain::esp_get_munic_siane(), *15*, *17*
mapSpain::esp_get_tiles, *33*
mapSpain::esp_get_tiles(), *32*, *34*

NULL, *3*, *5, 6*, *8, 9*, *11–13*, *15–17*, *19, 20*, *22*, *27*, *29*, *31*, *36*

regex, *15*

sf, *3*, *6*, *9*, *14*, *27*, *29*, *31–33*
sf::st_bbox(), *27*, *29*, *32*
sf::st_centroid(), *15*
sf::st_crs(), *26*
sfc, *27*, *29*, *31*, *33*
SpatRaster, *33*

tempdir(), *24*
terra::crop(), *33*
terra::mask(), *33*
terra::plotRGB(), *34*
terra::RGB(), *33, 34*
tibble, *5*, *8*, *11, 12*, *15–17*, *19, 20*, *22*, *25*
tidyterra::geom_spatraster_rgb(), *34*
tools::R_user_dir(), *13*, *24*