# Package ‘VicmapR’

June 20, 2023

**Title**  Access Victorian Spatial Data Through Web File Services (WFS)

**Version**  0.2.3


**License**  Apache License (== 2.0)

**Encoding**  UTF-8

**LazyData**  true

**RoxygenNote**  7.1.1


**BugReports**  [https://github.com/justincally/VicmapR/issues](https://github.com/justincally/VicmapR/issues)

**Suggests**  testthat (>= 2.1.0), covr, rmarkdown

**Imports**  magrittr, httr, sf (>= 0.8), dplyr, purrr, methods, cli, DBI, xml2, glue, dbplyr (>= 2.2.0), rlang, curl, rvest, lubridate, knitr, kableExtra, mapview, leaflet, stringr

**VignetteBuilder**  knitr

**SystemRequirements**  GDAL (>= 3.0.0), GEOS (>= 3.4.0), PROJ (>= 7.0.0)

**Collate**  'globals.R' 'build-query.R' 'check_geoserver.R'

'cql-predicates.R' 'cql-translate.R' 'filter.R' 'select.R'

'listFeatures.R' 'polygonFormat.R' 'utils-collect.R'

'utilis-classes.R' 'utils-head.R' 'utils-metadata.R'

'utilis-options.R' 'utils-pipe.R' 'utils-show_query.R'

'metadata.R' 'convert-layer-names.R' 'data.R'

**Depends**  R (>= 2.10)

**NeedsCompilation**  no
check_geoserver

Description

VicmapR relies upon a functioning geoserver. If for whatever reason the geoserver is not functioning then the functions in this package will not work. This function will check the response of the geoserver; erroring out if the connection is down.

Usage

check_geoserver(timeout = 15, quiet = FALSE)
convert_layer_name

Arguments

- **timeout** numeric: the time (in seconds) to wait for the response before timing out (default is 15)
- **quiet** logical: whether to silently check the connection and if working, return nothing. If FALSE (default), the status message will be printed (http_status)

Value

logical, TRUE if the geoserver is working

Examples

```r
try(
  check_geoserver()
)
```

---

convert_layer_name  Convert layer names from old platform naming convention to the new platform naming convention

Description

Convert layer names from old platform naming convention to the new platform naming convention

Usage

```r
convert_layer_name(x)
```

Arguments

- **x** object of class vicmap_promise

Value

object of class vicmap_promise

Examples

```r
try(
  vicmap_query(layer = "datavic:VMHYDRO_WATERCOURSE_DRAIN")
)
```
CQL Escaping

Description

Write a CQL expression to escape its inputs, and return a CQL/SQL object. Used when writing
filter expressions in `vicmap_query()`.

Usage

CQL(...)

Arguments

... Character vectors that will be combined into a single CQL statement.

Details

See the CQL/ECQL for Geoserver website.

The code for cql escaping was developed by the bcdata team: https://bcgov.github.io/bcdata/
reference/cql_geom_predicates.html

Value

An object of class c("CQL", "SQL")

Examples

CQL("FOO > 12 & NAME LIKE 'A&'")

cql_geom_predicates CQL Geometry Predicates

Description

Functions to construct a CQL expression to be used to filter results from `vicmap_query()`. See
the geoserver CQL documentation for details. The sf object is automatically simplified to a less
complex sf object to reduce the complexity of the Web Service call. Subsequent in-memory filtering
may be needed to achieve exact results.
Usage

EQUALS(geom)

DISJOINT(geom)

INTERSECTS(geom)

TOUCHES(geom)

CROSSES(geom)

WITHIN(geom)

CONTAINS(geom)

OVERLAPS(geom)

RELATE(geom, pattern)

BBOX(coords, crs = NULL)

DWITHIN(
    geom,
    distance,
    units = c("meters", "feet", "statute miles", "nautical miles", "kilometers")
)

BEYOND(
    geom,
    distance,
    units = c("meters", "feet", "statute miles", "nautical miles", "kilometers")
)

Arguments

geom an sf/sfc/sfg or bbox object (from the sf package)

pattern spatial relationship specified by a DE-9IM matrix pattern. A DE-9IM pattern is a string of length 9 specified using the characters *TF012. Example: '1*T***T**'

coords the coordinates of the bounding box as four-element numeric vector c(xmin, ymin, xmax, ymax), a bbox object from the sf package (the result of running sf::st_bbox() on an sf object), or an sf object which then gets converted to a bounding box on the fly.

crs (Optional) A numeric value or string containing an SRS code. If coords is a bbox object with non-empty crs, it is taken from that. (For example, 'EPSG:3005' or just 3005. The default is to use the CRS of the queried layer)

distance numeric value for distance tolerance
units

units that distance is specified in. One of "feet", "meters", "statute miles", "nautical miles", "kilometers"

Details

The code for these cql predicates was developed by the bcdata team: https://bcgov.github.io/bcdata/reference/cql_geom_predicates.html

Value

a CQL expression to be passed on to the WFS call

data_citation

Layer Metadata

Description

formatted metadata attributes of a given vicmap layer (vicmap_query(layer)). Metadata is retrieved from the Vicmap catalogue. data_citation() prints a BibTex style citation for a given record; similar to base::citation(). data_dictionary() returns a table with names, types and descriptions of the data within the selected layer (see details). get_metadata() returns a list with three elements, containing metadata, the data dictionary and the url of the metadata for the record.

Usage

data_citation(x = NULL, metadataID = NULL)
data_dictionary(x = NULL, metadataID = NULL)
get_metadata(x = NULL, metadataID = NULL)

Arguments

x

Object of class vicmap_promise (likely passed from vicmap_query())

metadataID

character: ID of data (useful if data is not available through WFS)

Value

citation, data.frame or list

Examples

try(
data_citation(vicmap_query(layer = "datavic:VMHYDRO_WATERCOURSE_DRAIN"))
)
try(
data_dictionary(vicmap_query(layer = "datavic:VMHYDRO_WATERCOURSE_DRAIN"))
)

try(
get_metadata(vicmap_query(layer = "datavic:VMHYDRO_WATERCOURSE_DRAIN"))
)

---

**feature_hits**  
*The Number of Rows of the Promised Data*

**Description**

`feature_hits()` returns an integer of the number of rows that match the passed query/promise. This is similar to how `nrow()` works for a data.frame, however it will evaluate the number of rows to be returned without having to download the data.

**Usage**

`feature_hits(x)`

**Arguments**

- `x`  
  object of class `vicmap_promise`

**Value**

integer

**Examples**

```r
vicmap_query(layer = "open-data-platform:hy_watercourse") %>%
  feature_hits()
```
**Description**

`geom_col_name` returns a single value for the name of the geometry column for the WFS layer selected in the `vicmap_promise` object (e.g. `SHAPE`). This column will become the geometry column when using `collect()`. `feature_cols()` provides a vector of all column names for the WFS layer selected in the `vicmap_promise` object and `get_col_df()` returns a data.frame with the column names and their XML schema string datatypes.

**Usage**

```r
geom_col_name(x)

feature_cols(x)

get_col_df(x)
```

**Arguments**

- `x`: object of class `vicmap_promise`

**Value**

- character/data.frame

**Examples**

```r
# Return the name of the geometry column
vicmap_query(layer = "open-data-platform:hy_watercourse") %>%
  geom_col_name()

# Return the column names as a character vector
vicmap_query(layer = "open-data-platform:hy_watercourse") %>%
  feature_cols()

# Return a data.frame of the columns and their XML schema string datatypes
try(
  vicmap_query(layer = "open-data-platform:hy_watercourse") %>%
    get_col_df()
)
```
**listLayers**

*List Available WFS Layers*

**Description**

Lists layers available from the WFS geoserver. This is similar to sending the WFS request of `getFeatureTypes`. `listLayers()` returns a data.frame with the 'Name' and title of the layers available. The 'Name' is what is used within `vicmap_query()` while the title provides somewhat of a description/clarification about the layer.

**Usage**

```r
listLayers(..., abstract = TRUE)
```

**Arguments**

- `...`: Additional arguments passed to `grep`. The pattern argument can be used to search for specific layers with matching names or titles.
- `abstract`: Whether to return a column of abstract (and metadata ID), the default is true. Switching to FALSE will provide a data.frame with only 2 columns and may be slightly faster.

**Value**

data.frame of 2 (abstract = FALSE) or 4 (abstract = TRUE) columns

**Examples**

```r
try(
  listLayers(pattern = "trees", ignore.case = TRUE)
)
```

**name_conversions**

*Name conversions between old and new geoserver*

**Description**

A dataset containing the names of the old and corresponding new data on geoserver, including relevant CQL filters

**Usage**

data(name_conversions)
Format

A data frame with 630 rows and 5 variables

Details

- **Original_Layer_Name.** The name of the original data 'typeNames', without the 'datavic:' prefix
- **New_Layer_Name.** The new 'typeNames' for the corresponding data stored on the AWS cloud geoserver
- **CQL_FILTER.** The CQL filter that needs to be applied to the new data to match the old dataset
- **full_original_name.** The full name of the original layer (with prefix)
- **full_new_name.** The full name of the new data (with prefix to be used when calling the layer with vicmap_query())

---

**print.vicmap_promise**  
*Print a Snapshot of the Data*

Description

print() displays a cut of the data (no more than six rows) alongside the number of rows and columns that would be returned.

Usage

```r
## S3 method for class 'vicmap_promise'
print(x, ...)
```

Arguments

- **x**  
  object of class vicmap_promise (likely passed from vicmap_query())
- **...**  
  arguments to be passed to print

Value

vicmap_promise (invisible), promise sample printed to console

Examples

```r
try(
  query <- vicmap_query(layer = "open-data-platform:hy_watercourse")
)
try(
  print(query)
)
```
vicmap_options

vicmap_options  options

Description

This function retrieves bcdata specific options that can be set. These options can be set using 
option({name of the option} = {value of the option}). The default options are purposefully 
set conservatively to hopefully ensure successful requests. Resetting these options may result in 
failed calls to the data catalogue. Options in R are reset every time R is re-started.

vicmap.max_geom_pred_size is the maximum size of an object used for a geometric operation. Objects that are bigger than this value will be simplified in the request call using sf::st_simplify(). This is done to reduce the size of the query being sent to the WFS geoserver.

vicmap.chunk_limit is an option useful when dealing with very large data sets. When requesting large objects from the catalogue, the request is broken up into smaller chunks which are then re-combined after they’ve been downloaded. VicmapR does this all for you but using this option you can set the size of the chunk requested. On faster internet connections, a bigger chunk limit could be useful while on slower connections, it is advisable to lower the chunk limit. Chunks must be less than 70000.

vicmap.base_url is the base wfs url used to query the geoserver.

vicmap.backend is the backend software running the geoserver. The data migration in 2022/2023 will change the backend from ‘Oracle’ to ‘AWS’

Usage

vicmap_options()

check_chunk_limit()

Value

vicmap_options() returns a data.frame

Examples

vicmap_options()

vicmap_query  Establish Vicmap Query

Description

Begin a Vicmap WFS query by selecting a WFS layer. The record must be available as a Web Feature Service (WFS) layer (listed in listLayers())
Usage

```r
vicmap_query(layer, CRS = 4283, wfs_version = "2.0.0")
```

Arguments

- `layer`\footnote{vicmap layer to query. Options are listed in `listLayers()`.}
- `CRS`\footnote{Coordinate Reference System (default is 4283).}
- `wfs_version`\footnote{The current version of WFS is 2.0.0. GeoServer supports versions 2.0.0, 1.1.0, and 1.0.0. However in order for filtering to be correctly applied `wfs_version` must be 2.0.0 (default is 2.0.0).}

Details

The returned `vicmap_promise` object is not data, rather it is a ‘promise’ of the data that can be returned if `collect()` is used; which returns an `sf` object.

Value

object of class `vicmap_promise`, which is a ‘promise’ of the data that can be returned if `collect()` is used

Examples

```r
try(
  vicmap_query(layer = "open-data-platform:hy_watercourse")
)
```
Index

* datasets
  name_conversions, 9
BBOX (cql_geom_predicates), 4
BEYOND (cql_geom_predicates), 4

check_chunk_limit (vicmap_options), 11
check_geoserver, 2
CONTAINS (cql_geom_predicates), 4
convert_layer_name, 3
CQL, 4
cql_geom_predicates, 4
CROSSES (cql_geom_predicates), 4

data_citation, 6
data_dictionary (data_citation), 6
DISJOINT (cql_geom_predicates), 4
DWITHIN (cql_geom_predicates), 4

EQUALS (cql_geom_predicates), 4

feature_cols (geom_col_name), 8
feature_hits, 7

geom_col_name, 8
get_col_df (geom_col_name), 8
get_metadata (data_citation), 6
grep, 9

http_status, 3
INTERSECTS (cql_geom_predicates), 4
listLayers, 9
name_conversions, 9

OVERLAPS (cql_geom_predicates), 4

print, 10
print.vicmap_promise, 10

RELATE (cql_geom_predicates), 4
TOUCHES (cql_geom_predicates), 4
vicmap_options, 11
vicmap_query, 11
vicmap_query(), 4, 6, 10
WITHIN (cql_geom_predicates), 4