# Package 'rorcid'

June 7, 2019

**Title** Interface to the 'Orcid.org' API

**Description** Client for the 'Orcid.org' API (<https://orcid.org/>). Functions included for searching for people, searching by 'DOI', and searching by 'Orcid' 'ID'.

**Version** 0.5.0

**License** MIT + file LICENSE

**URL** https://github.com/ropensci/rorcid

**BugReports** https://github.com/ropensci/rorcid/issues

**LazyLoad** yes

**LazyData** true

**Encoding** UTF-8

**Depends** R (>= 3.5.0)

**Imports** crul (>= 0.7.4), httr, fauxpas (>= 0.2.0), jsonlite (>= 1.6), xml2 (>= 1.2.0), tibble (>= 2.1.3), data.table

**Suggests** testthat, knitr, rcrossref, handlr, httpuv

**RoxygenNote** 6.1.1

**VignetteBuilder** knitr

**X-schema.org-applicationCategory** Literature

**X-schema.org-keywords** identifiers, literature, publications, citations, scholarly, people

**X-schema.org-isPartOf** https://ropensci.org

**NeedsCompilation** no

**Author** Scott Chamberlain [aut, cre] (<https://orcid.org/0000-0003-1444-9135>), rOpenSci [fnd] (https://ropensci.org/)

**Maintainer** Scott Chamberlain <myrmecocystus@gmail.com>

**Repository** CRAN

**Date/Publication** 2019-06-07 18:50:04 UTC

# R topics documented:

`rorcid-package` *A programmatic R interface the Orcid.org API*

## Description

A R interface to the Orcid public API. **rorcid** is not a product developed or distributed by ORCID.

ORCID website: https://orcid.org/

Orcid API docs: http://members.orcid.org/api

Some key **rorcid** function:

- `as.orcid()` - coerce various inputs to ORCID class
- `browse()` - browse to a profile in your default browser
- `check_dois()` - check that strings are likely to be DOIs
- `identifiers()` - grab identifiers out of various objects
- `orcid()` and `orcid_search()` - Search for ORCID id's
- `orcid_doi()` - Search by DOI
- `orcid_id()` - Search by ORCID id, and get either bio, profile, or works
- `works()` - Parse out works from various objects

## API routes not implemented

Not quite sure what these do so haven't messed with them.

- `/{orcid}/notification-permission/{id}`
- `/client/{client_id}`
- `/group-id-record`
- `/group-id-record/{putCode}`

## Rate Limits

Definitions:

- Request a second - Number of request that can be made a second. Value: 8 per second (24 with API v2rc+) - Haven't been able to find up to date values for API v3 (so assume they are the same I guess)
- Burst - Number of request we will allow to be queued before rejecting. The request in the queue are slowed down at the request a second rate. Value: 40 (same with API v2rc+) - Haven't been able to find up to date values for API v3 (so assume they are the same I guess)

If you exceed the burst, you'll get a 503 responses. Developers should do their best to avoid approaching those limits.

## Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

## See Also

rorcid-auth for Authentication information

---

`as.orcid` *Convert an ORCID or something like an ORCID object*

---

## Description

Convert an ORCID or something like an ORCID object

## Usage

```
as.orcid(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | An ORCID id, passed to `print` |
| `...` | Further args passed on to `orcid_id()` |

## Value

an S3 object of class `or_cid`, which pretty prints for brevity

## Examples

```
## Not run:
as.orcid(x="0000-0002-1642-628X")
out <- orcid("text:English", rows = 20)
as.orcid(out$`orcid-identifier.path`[1])

# Passon further args to orcid_id()
as.orcid("0000-0002-1642-628X", verbose = TRUE)

# Browse to a profile
# browse(as.orcid("0000-0002-1642-628X"))

# many ORCIDs as a character vector
ids <- c("0000-0002-1642-628X", "0000-0002-9341-7985")
as.orcid(ids)

# many in a list via orcid_id()
(x <- lapply(ids, orcid_id))
as.orcid(x)

## End(Not run)
```

## browse *Navigate to an ORCID profile in your default browser*

### Description

Navigate to an ORCID profile in your default browser

### Usage

```
browse(orcid)
```

### Arguments

| | |
|---|---|
| orcid | An or_cid class object |

### Examples

```
## Not run:
browse(as.orcid("0000-0002-1642-628X"))

## End(Not run)
```

## check_dois *Verify DOI's are likely good*

### Description

Verify DOI's are likely good

### Usage

```
check_dois(x)
```

### Arguments

| | |
|---|---|
| x | One or more DOIs |

### Value

A list of length two, one slot for good DOIs, one for bad

## Examples

```
## Not run:
check_dois("10.1087/20120404")

dois=c("10.1371/journal.pone.0025995","10.1371/journal.pone.0053712",
       "10.1371/journal.pone.0054608","10.1371/journal.pone.0055937")
check_dois(dois)

dois=c("10.1016/j.medpal.2008.12.005","10.1080/00933104.2000.10505926",
       "10.1037/a0024480", "10.1002/anie.196603172","2344","asdf","232",
       "asdf","23dd")
check_dois(dois)

## End(Not run)
```

---

`fields` *Lookup table for search fields*

---

## Description

Lookup table for search fields

---

`identifiers` *Get identifiers*

---

## Description

This function aims to pluck out just identifiers into a vector for easy use downstream (e.g., use DOIs to fetch article metadata). You can still manually fetch additional data from outputs of functions in this package.

## Usage

```
identifiers(x, type = "doi", ...)

## S3 method for class 'works'
identifiers(x, type = "doi", ...)

## S3 method for class 'list'
identifiers(x, type = "doi", ...)

## S3 method for class 'orcid_id'
identifiers(x, type = "doi", ...)

## S3 method for class 'orcid'
identifiers(x, type = "doi", ...)
```

```
## S3 method for class 'orcid_doi'
identifiers(x, type = "doi", ...)
```

## Arguments

| | |
|---|---|
| x | An object of class works, orcid, orcid_id, orcid_doi, or a list that contains any number of the previous objects. |
| type | (character) One of doi (default), pmid, pmc, eid, other_id, orcid, scopus, researcherid. The orcid's here are for works, not individuals. This parameter is ignored for classes `orcid` and `orcid_doi` both of which would go down a rabbit hole of getting works for all ORCIDs which could take a while. |
| ... | Ignored. |

## Value

(character) vector of identifiers, or NULL if none found

## References

list of identifiers https://pub.qa.orcid.org/v2.0/identifiers?locale=en

## Examples

```
## Not run:
# Result of call to works()
x <- works(orcid_id("0000-0001-8607-8025"))
# doi by default
identifiers(x)
# orcids
identifiers(x, "orcid")
# pmid
identifiers(x, "pmid")
# pmc
identifiers(x, "pmc")
# other_id
identifiers(x, "other_id")

# Result of call to orcid_id()
x <- orcid_id(orcid = "0000-0002-9341-7985")
identifiers(x, "doi")
identifiers(x, "eid")

# Result of call to orcid()
x <- orcid(query="carl+boettiger")
identifiers(x)

# Result of call to orcid_doi()
x <- orcid_doi(dois="10.1087/20120404", fuzzy=TRUE)
identifiers(x)
```

```
## End(Not run)
```

---

`issn_title` *Lookup vector for journal titles by ISSN*

---

## Description

named vector of journal titles. the values are journal titles and the names are ISSN's.

## Details

length: 57,968

data collected on 2018-06-13 from Crossref

---

`orcid` *Search for ORCID ID's.*

---

## Description

Search for ORCID ID's.

## Usage

```
orcid(query = NULL, start = NULL, rows = NULL, recursive = FALSE,
  defType = NULL, q.alt = NULL, qf = NULL, mm = NULL, qs = NULL,
  pf = NULL, ps = NULL, pf2 = NULL, ps2 = NULL, pf3 = NULL,
  ps3 = NULL, tie = NULL, bq = NULL, bf = NULL, boost = NULL,
  uf = NULL, lowercaseOperators = NULL, fuzzy = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `query` | Search terms. You can do quite complicated queries using the SOLR syntax. See examples below. For all possible fields to query, do `data(fields)` |
| `start` | Result number to start on. Keep in mind that pages start at 0. Default: 0 |
| `rows` | Numer of results to return. Default: 10. Max: 200 |
| `recursive` | Keep drilling down until all records are retrieved for the given query, default FALSE (logical). If `recursive=TRUE`, rows and start parameters are ignored. |
| `defType` | Query syntax. One of edismax or X. See Details for more. |
| `q.alt` | If specified, this query will be used (and parsed by default using standard query parsing syntax) when the main query string is not specified or blank. This comes in handy when you need something like a match-all-docs query (don't forget &rows=0 for that one!) in order to get collection-wise faceting counts. |
| `qf` | (Query Fields) List of fields and the "boosts" to associate with each of them when building DisjunctionMaxQueries from the user's query |

mm (Minimum 'Should' Match) See the wiki here http://wiki.apache.org/solr/ExtendedDisMax#mm_.28Minimum_.27Should.27_Match.29

qs (Query Phrase Slop) Amount of slop on phrase queries explicitly included in the user's query string (in qf fields; affects matching).

pf (Phrase Fields) Once the list of matching documents has been identified using the "fq" and "qf" params, the "pf" param can be used to "boost" the score of documents in cases where all of the terms in the "q" param appear in close proximity. Read more here http://wiki.apache.org/solr/ExtendedDisMax#pf_.28Phrase_Fields.29

ps (Phrase Slop) Default amount of slop on phrase queries built with "pf", "pf2" and/or "pf3" fields (affects boosting).

pf2 (Phrase bigram fields) As with 'pf' but chops the input into bi-grams, e.g. "the brown fox jumped" is queried as "the brown" "brown fox" "fox jumped"

ps2 (Phrase bigram slop) As with 'ps' but sets default slop factor for 'pf2'. If not specified, 'ps' will be used.

pf3 (Phrase trigram fields) As with 'pf' but chops the input into tri-grams, e.g. "the brown fox jumped" is queried as "the brown fox" "brown fox jumped"

ps3 (Phrase trigram slop) As with 'ps' but sets default slop factor for 'pf3'. If not specified, 'ps' will be used.

tie (Tie breaker) Float value to use as tiebreaker in DisjunctionMaxQueries (should be something much less than 1). Read more here http://wiki.apache.org/solr/ExtendedDisMax#tie_.28Tie_breaker.29

bq (Boost Query) A raw query string (in the SolrQuerySyntax) that will be included with the user's query to influence the score. Read more here http://wiki.apache.org/solr/ExtendedDisMax#bq_.28Boost_Query.29

bf (Boost Function, additive) Functions (with optional boosts) that will be included in the user's query to influence the score. Any function supported natively by Solr can be used, along with a boost value, e.g.: recip(rord(myfield),1,2,3)^1.5. Read more here http://wiki.apache.org/solr/ExtendedDisMax#bf_.28Boost_Function.2C_additive.29

boost (Boost Function, multiplicative) As for 'bf' but multiplies the boost into the score

uf (User Fields) Specifies which schema fields the end user shall be allowed to query for explicitly. This parameter supports wildcards. Read more here http://wiki.apache.org/solr/ExtendedDisMax#uf_.28User_Fields.29

lowercaseOperators
This param controls whether to try to interpret lowercase words as boolean operators such as "and", "not" and "or". Set &lowercaseOperators=true to allow this. Default is "false".

fuzzy Use fuzzy matching on input DOIs. Defaults to FALSE. If FALSE, we stick "digital-object-ids" before the DOI so that the search sent to ORCID is for that exact DOI. If TRUE, we use some regex to find the DOI.

... Curl options passed on to crul::HttpClient()

## Details

All query syntaxes available in SOLR 3.6 ( https://cwiki.apache.org/confluence/display/solr/The+Standard+Query+Parser) are supported, including Lucene with Solr extensions (default), DisMax, and Extended Dismax.

You can use any of the following within the query statement: given-names, family-name, credit-name, other-names, email, grant-number, patent-number, keyword, worktitle, digital-objectids, current-institution, affiliation-name, current-primary-institution, text, or past-institution.

For more complicated queries the ORCID API supports using ExtendedDisMax. See the documentation on the web here: http://wiki.apache.org/solr/ExtendedDisMax

Note that when constructing queries, you don't need to use syntax like +, etc., `crul`, the http client we use internally, will do that for you. For example, instead of writing `johnson+cardiology`, just write `johnson cardiology`, and instead of writing `johnson+AND+cardiology`, write `johnson AND cardiology`. Though, you still need to use `AND`, `OR`, etc. to join term/queries together.

## Value

a data.frame (tibble). You can access number of results found like `attr(result, "found")`. Note that with ORCID API v2 and greater, results here are only the identifiers. To get other metadata/data you can take the identifiers and use other functions in this package.

## References

https://members.orcid.org/api/tutorial/search-orcid-registry

## See Also

`orcid_doi()` `orcid_id()` `orcid_search()`

## Examples

```
## Not run:
# Get a list of names and Orcid IDs matching a name query
orcid(query="carl+boettiger")
orcid(query="given-names:carl AND family-name:boettiger")

# by email
orcid(query="email:cboettig@berkeley.edu")

# You can string together many search terms
orcid(query="johnson cardiology houston")

# And use boolean operators
orcid("johnson AND(caltech OR 'California Institute of Technology')")

# And you can use start and rows arguments to do pagination
orcid("johnson cardiology houston", start = 2, rows = 3)

# Use search terms, here family name
orcid("family-name:Sanchez", start = 4, rows = 6)
```

```
# Use search terms, here...
orcid(query="Raymond", start=0, rows=10, defType="edismax")

# Search using keywords
orcid(query="keyword:ecology")

# Search by DOI
orcid(query="10.1087/20120404")

# Note the difference between the first wrt the second and third
## See also orcid_doi() function for searching by DOIs
orcid("10.1087/20120404")
orcid('"10.1087/20120404"')
## doi
orcid('digital-object-ids:"10.1087/20120404"')
## doi prefix
orcid('digital-object-ids:"10.1087/*"')

# search by work titles
orcid('work-titles:Modern developments in holography and its materials')
orcid('pmc:PMC3901677')

## Using more complicated SOLR queries

# Use the qf parameter to "boost" query fields so they are ranked higher
#  See how it is different than the second query without using "qf"
orcid(defType = "edismax", query = "Raymond",
   qf = "given-names^1.0 family-name^2.0", start = 0, rows = 10)
orcid(query = "Raymond", start = 0, rows = 10)

# Use other SOLR parameters as well, here mm. Using the "mm" param, 1 and
# 2 word queries require that all of the optional clauses match, but for
# queries with three or more clauses one missing clause is allowed...
# See for more: http://bit.ly/1uyMLDQ
orcid(defType = "edismax",
      query="keyword:ecology OR evolution OR conservation",
      mm = 2, rows = 20)

## End(Not run)
```

---

`orcid_activities` *Get activities for a person*

---

## Description

Get activities for a person

## Usage

```
orcid_activities(orcid, ...)
```

### Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| ... | Curl options passed on to crul::HttpClient() |

### Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

### Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

### Examples

```
## Not run:
res <- orcid_activities(orcid = "0000-0002-9341-7985")
res$`0000-0002-9341-7985`
names(res$`0000-0002-9341-7985`)
res$`0000-0002-9341-7985`$`last-modified`
res$`0000-0002-9341-7985`$`educations`
res$`0000-0002-9341-7985`$`fundings`
res$`0000-0002-9341-7985`$`peer-reviews`
res$`0000-0002-9341-7985`$`works`

## End(Not run)
```

---

## orcid_address *Get address information for a person*

---

### Description

Get address information for a person

### Usage

```
orcid_address(orcid, put_code = NULL, format = "application/json", ...)
```

### Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| ... | Curl options passed on to crul::HttpClient() |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
# all addresses
res <- orcid_address(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
names(res$`0000-0002-1642-628X`)
res$`0000-0002-1642-628X`$`address`

# individual address
orcid_address(orcid = "0000-0002-1642-628X", 288064)

# format
orcid_address(orcid = "0000-0002-1642-628X", 288064, "application/xml")

## End(Not run)
```

---

# orcid_auth *ORCID authorization*

---

## Description

ORCID authorization

## Usage

```
orcid_auth(scope = "/authenticate", reauth = FALSE,
  redirect_uri = getOption("rorcid.redirect_uri"))
```

## Arguments

| | |
|---|---|
| scope | (character) one or more scopes. default: "/authenticate" |
| reauth | (logical) Force re-authorization? |
| redirect_uri | (character) a redirect URI. optional |

## Details

There are two ways to authorise with **rorcid**:

- Use a token as a result of a OAuth authentication process. The token is a alphanumeric UUID, e.g. `dc0a6b6b-b4d4-4276-bc89-78c1e9ede56e`. You can get this token by running `orcid_auth()`, then storing that key (the uuid alone, not the "Bearer " part) either as en environment variable in your `.Renviron` file in your home directory (with the name `ORCID_TOKEN`), or as an R option in your `.Rprofile` file (with the name `orcid_token`). See Startup for more information. Either an environment variable or R option work. If we don't find either we do the next option.
- Interactively login with OAuth. This doesn't require any input on your part. We use a client id and client secret key to ping ORCID.org; at which point you log in with your username/password; then we get back a token (same as the above option). We don't know your username or password, only the token that we get back. We cache that token locally in a hidden file in whatever working directory you're in. If you delete that file, or run the code from a new working directory, then we re-authorize.

We recommend the former option. That is, get a token and store it as an environment variable.

If both options above fail, we proceed without using authentication. ORCID does not require authentication at this point, but may in the future - this prepares you for when that happens :)

## Value

a character string with the access token prefixed with "Bearer "

## ORCID OAuth Scopes

See `https://members.orcid.org/api/orcid-scopes` for more

## Computing environments without browsers

One pitfall is when you are using rorcid on a server, and you're ssh'ed in, so that there's no way to open a browser to do the OAuth browser flow. Similarly for any other situation in which a browser can not be opened. In this case, run `orcid_auth()` on another machine in which you do have the ability to open a browser, then collect the info that's ouptput from `orcid_auth()` and store it as an environment variable (see above).

## Examples

```
## Not run:
x <- orcid_auth()
orcid_auth(reauth = TRUE)
#orcid_auth(scope = "/read-public", reauth = TRUE)

## End(Not run)
```

`orcid_bio` *Get biography data for a person*

## Description

Get biography data for a person

## Usage

```
orcid_bio(orcid, format = "application/json", ...)
```

## Arguments

| | |
|---|---|
| `orcid` | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| `format` | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| `...` | Curl options passed on to `crul::HttpClient()` |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
res <- orcid_bio(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
res$`0000-0002-1642-628X`$`created-date`
res$`0000-0002-1642-628X`$`last-modified-date`
res$`0000-0002-1642-628X`$`content`
res$`0000-0002-1642-628X`$`visibility`
res$`0000-0002-1642-628X`$path

## End(Not run)
```

`orcid_citations` *Get citations*

## Description

Get citations

## Usage

```
orcid_citations(orcid, put_code = NULL, cr_format = "bibtex",
  cr_style = "apa", cr_locale = "en-US", ...)
```

## Arguments

| | |
|---|---|
| `orcid` | (character) Orcid identifier(s) of the form XXXX-XXXX-XXXX-XXXX. required. |
| `put_code` | (character/integer) one or more put codes. up to 50. optional |
| `cr_format` | Used in Crossref queries only. Name of the format. One of "rdf-xml", "turtle", "citeproc-json", "citeproc-json-ish", "text", "ris", "bibtex" (default), "crossref-xml", "datacite-xml","bibentry", or "crossref-tdm". The format "citeproc-json-ish" is a format that is not quite proper citeproc-json. passed to `rcrossref::cr_cn`. The special "citeproc2bibtex" value asks for citeproc-json from Crossref, then converts it into bibtex format using handlr::HandlrClient |
| `cr_style` | Used in Crossref queries only. A CSL style (for text format only). See 'get_styles()' for options. Default: apa. passed to `rcrossref::cr_cn` |
| `cr_locale` | Used in Crossref queries only. Language locale. See Sys.getlocale, passed to `rcrossref::cr_cn` |
| `...` | Curl options passed on to crul::HttpClient |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

This function is focused on getting citations only. You can get all citations for an ORCID, or for certain works using a PUT code, or for many PUT codes.

We attempt to get citations via Crossref using **rcrossref** whenever possible as they are the most flexible and don't have as many mistakes in the text. If there is no DOI, we fetch the citation from ORCID.

Right now we get JSON citations back. We'd like to support bibtex format. DOI.org supports this but not ORCID.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

data.frame, with the columns:

- put: ORCID PUT code, identifying the work identifier in ORCID's records
- id: the external identifier
- id_type: the type of external identifier
- format: the citation format retrieved
- citation: the citation as JSON

## Examples

```
## Not run:
(res <- orcid_citations(orcid = "0000-0002-9341-7985"))
(res2 <- orcid_citations(orcid = "0000-0002-1642-628X"))
(res2 <- orcid_citations(orcid = c("0000-0002-9341-7985", "0000-0002-1642-628X")))

# get individual works
## a single put code
(a <- orcid_citations(orcid = "0000-0002-9341-7985", put_code = 5011717))
## many put codes
(b <- orcid_citations(orcid = "0000-0002-9341-7985",
   put_code = c(5011717, 15536016)))

# request other formats, Crossref only
orcid_citations(orcid = "0000-0002-9341-7985", cr_format = "turtle")

# parse citation data if you wish
# for parsing bibtex can use bibtex package or others
(res <- orcid_citations(orcid = "0000-0002-9341-7985"))
lapply(res[res$format == "csl-json", "citation"][[1]], jsonlite::fromJSON)

# lots of citations
orcid_citations(orcid = "0000-0001-8642-6325")

# example with no external identifier, returns NA's
orcid_citations(orcid = "0000-0001-8642-6325", 26222265)

## End(Not run)
```

---

# orcid_distinctions *Get distinction for a person*

---

## Description

Get distinction for a person

### Usage

```
orcid_distinctions(orcid, put_code = NULL, format = "application/json",
  summary = FALSE, ...)
```

### Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| summary | (logical) get education summary for a put code. Default: FALSE |
| ... | Curl options passed on to crul::HttpClient() |

### Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

### Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

### Examples

```
## Not run:
res <- orcid_distinctions(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
res$`0000-0002-1642-628X`$`created-date`
res$`0000-0002-1642-628X`$`affiliation-group`
res$`0000-0002-1642-628X`$path

## End(Not run)
```

---

## orcid_doi *Search for ORCID ID's using DOIs*

---

### Description

Search for ORCID ID's using DOIs

### Usage

```
orcid_doi(dois = NULL, start = NULL, rows = NULL, fuzzy = FALSE,
  ...)
```

### Arguments

| | |
|---|---|
| dois | (character) Digital object identifier (DOI), a vector fo DOIs. |
| start | (integer) Result number to start on. Keep in mind that pages start at 0. |
| rows | (integer) Numer of results to return. |
| fuzzy | (logical) Use fuzzy matching on input DOIs. Defaults to FALSE. If FALSE, we stick "digital-object-ids" before the DOI so that the search sent to ORCID is for that exact DOI. If TRUE, we use some regex to find the DOI. |
| ... | Curl options passed on to crul::HttpClient() |

### Examples

```
## Not run:
orcid_doi(dois="10.1087/20120404", fuzzy=TRUE)

# fuzzy is FALSE by default
orcid_doi(dois="10.1087/20120404", fuzzy=FALSE)

# This DOI is not a real one, but a partial DOI, then we can fuzzy search
# get more than defualt 10 records (or rows)
orcid_doi(dois="10.1087/2", fuzzy=TRUE, rows=20)

# If you don't input proper DOIs, the function will get mad
dois <- c("10.1371/journal.pone.0025995","10.1371/journal.pone.0053712",
       "10.1371/journal.pone.0054608","10.1371/journal.pone.0055937")
orcid_doi(dois=dois)

# dois <- c("10.1016/j.medpal.2008.12.005","10.1080/00933104.2000.10505926",
#          "10.1037/a0024480", "10.1002/anie.196603172","2344","asdf","232",
#          "asdf","23dd")
# orcid_doi(dois=dois)

orcid_doi(dois="10.1087/20120404", fuzzy=FALSE)
orcid_doi(dois="10.1371/journal.pone.0025995", fuzzy=FALSE)

## End(Not run)
```

---

## orcid_educations *Get education information for a person*

---

### Description

Get education information for a person

### Usage

```
orcid_educations(orcid, put_code = NULL, format = "application/json",
  summary = FALSE, ...)
```

### Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| summary | (logical) get education summary for a put code. Default: FALSE |
| ... | Curl options passed on to crul::HttpClient() |

### Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

### Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

### Examples

```
## Not run:
# all education data
res <- orcid_educations(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
names(res$`0000-0002-1642-628X`)
res$`0000-0002-1642-628X`$`education-summary`

# individual education records
orcid_educations(orcid = "0000-0002-1642-628X", 148494)

# education summary information
orcid_educations(orcid = "0000-0002-1642-628X", 148494, summary = TRUE)

## End(Not run)
```

---

## orcid_email *Get education information for a person*

---

### Description

Get education information for a person

### Usage

```
orcid_email(orcid, ...)
```

### Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| ... | Curl options passed on to crul::HttpClient() |

### Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

### Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

### Examples

```
## Not run:
res <- orcid_email(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
names(res$`0000-0002-1642-628X`)
res$`0000-0002-1642-628X`$`email`

## End(Not run)
```

---

## orcid_employments *Get employment information for a person*

---

### Description

Get employment information for a person

### Usage

```
orcid_employments(orcid, put_code = NULL, format = "application/json",
  summary = FALSE, ...)
```

### Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| summary | (logical) get employment summary for a put code. Default: FALSE |
| ... | Curl options passed on to crul::HttpClient() |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
# all employment data
res <- orcid_employments(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
names(res$`0000-0002-1642-628X`)
res$`0000-0002-1642-628X`$`employment-summary`

# individual employment records
orcid_employments(orcid = "0000-0002-1642-628X", 1115445)
orcid_employments(orcid = "0000-0002-1642-628X", 148496)

# employment summary information
orcid_employments(orcid = "0000-0002-1642-628X", 1115445, summary = TRUE)

## End(Not run)
```

---

# orcid_external_identifiers

*Get education information for a person*

---

## Description

Get education information for a person

## Usage

```
orcid_external_identifiers(orcid, put_code = NULL,
  format = "application/json", ...)
```

## Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| ... | Curl options passed on to `crul::HttpClient()` |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
# all data
res <- orcid_external_identifiers(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
names(res$`0000-0002-1642-628X`)
res$`0000-0002-1642-628X`$`external-identifier`

# individual records
orcid_external_identifiers(orcid = "0000-0002-1642-628X", 141736)

## End(Not run)
```

---

# orcid_fundings *Get funding information for a person*

---

## Description

Get funding information for a person

## Usage

```
orcid_fundings(orcid, put_code = NULL, format = "application/json",
  summary = FALSE, ...)
```

## Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| summary | (logical) get funding summary for a put code. Default: FALSE |
| ... | Curl options passed on to crul::HttpClient() |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
# all funding data
res <- orcid_fundings(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
names(res$`0000-0002-1642-628X`)
res$`0000-0002-1642-628X`$`group`

# individual funding records
orcid_fundings(orcid = "0000-0002-1642-628X", 385627)

# funding summary information
orcid_fundings(orcid = "0000-0002-1642-628X", 385627, summary = TRUE)

## End(Not run)
```

---

# orcid_id *Get data for particular ORCID's*

---

## Description

Get data for particular ORCID's

## Usage

```
orcid_id(orcid, ...)
```

## Arguments

| | |
|---|---|
| orcid | (character) A single Orcid identifier, of the form XXXX-XXXX-XXXX-XXXX |
| ... | Curl options passed on to crul::HttpClient() |

## Value

A named list of results - from a call to orcid_person()

## Examples

```
## Not run:
res <- orcid_id(orcid = "0000-0002-9341-7985")
res$`0000-0002-9341-7985`
res$`0000-0002-9341-7985`$`name`
res$`0000-0002-9341-7985`$`other-names`
res$`0000-0002-9341-7985`$`biography`
res$`0000-0002-9341-7985`$`researcher-urls`
res$`0000-0002-9341-7985`$`emails`
res$`0000-0002-9341-7985`$`addresses`
res$`0000-0002-9341-7985`$`keywords`
res$`0000-0002-9341-7985`$`external-identifiers`
res$`0000-0002-9341-7985`$`emails`

ids <- c("0000-0003-1620-1408", "0000-0002-9341-7985")
res <- lapply(ids, orcid_id)
vapply(res, function(x) x[[1]]$name$`family-name`$value, "")

## End(Not run)
```

---

# orcid_invited_positions

*Get invited positions for a person*

---

## Description

Get invited positions for a person

## Usage

```
orcid_invited_positions(orcid, put_code = NULL,
  format = "application/json", summary = FALSE, ...)
```

## Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| summary | (logical) get education summary for a put code. Default: FALSE |
| ... | Curl options passed on to crul::HttpClient() |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
res <- orcid_invited_positions(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
res$`0000-0002-1642-628X`$`created-date`
res$`0000-0002-1642-628X`$`affiliation-group`
res$`0000-0002-1642-628X`$path

## End(Not run)
```

---

# orcid_keywords *Get education information for a person*

---

## Description

Get education information for a person

## Usage

```
orcid_keywords(orcid, put_code = NULL, format = "application/json",
  ...)
```

## Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| ... | Curl options passed on to `crul::HttpClient()` |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
# all data
res <- orcid_keywords(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
names(res$`0000-0002-1642-628X`)
res$`0000-0002-1642-628X`$`keyword`

# individual ones
orcid_keywords("0000-0002-1642-628X", 31202)

## End(Not run)
```

---

# orcid_memberships *Get memberships for a person*

---

## Description

Get memberships for a person

## Usage

```
orcid_memberships(orcid, put_code = NULL, format = "application/json",
  summary = FALSE, ...)
```

## Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| summary | (logical) get education summary for a put code. Default: FALSE |
| ... | Curl options passed on to crul::HttpClient() |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
res <- orcid_memberships(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
res$`0000-0002-1642-628X`$`created-date`
res$`0000-0002-1642-628X`$`affiliation-group`
res$`0000-0002-1642-628X`$memberships

## End(Not run)
```

---

# orcid_other_names *Get education information for a person*

---

## Description

Get education information for a person

## Usage

```
orcid_other_names(orcid, put_code = NULL, format = "application/json",
  ...)
```

## Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| ... | Curl options passed on to crul::HttpClient() |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
# all data
res <- orcid_other_names(orcid = "0000-0001-7893-4389")
res$`0000-0001-7893-4389`
names(res$`0000-0001-7893-4389`)
res$`0000-0001-7893-4389`$`other-name`

# individual ones
orcid_other_names("0000-0001-7893-4389", 239534)

# formats
orcid_other_names("0000-0001-7893-4389", format = "application/xml")

## End(Not run)
```

---

# orcid_peer_reviews *Get peer review information for a person*

---

## Description

Get peer review information for a person

## Usage

```
orcid_peer_reviews(orcid, put_code = NULL, format = "application/json",
  summary = FALSE, ...)
```

## Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| summary | (logical) get peer review summary for a put code. Default: FALSE |
| ... | Curl options passed on to crul::HttpClient() |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
# all peer review data
res <- orcid_peer_reviews(orcid = "0000-0001-7678-8656")
res$`0000-0001-7678-8656`
names(res$`0000-0001-7678-8656`)
res$`0000-0001-7678-8656`$`group`

# get individual works
orcid_peer_reviews("0000-0003-1444-9135", 75565)

# summary
orcid_peer_reviews("0000-0003-1444-9135", 75565, summary = TRUE)

# get Journal titles via ISSN's provided in results, using the
# provided issn_title dataset
x <- orcid_peer_reviews("0000-0001-7678-8656", put_code = "220419")
issn <- strsplit(x[[1]]$`review-group-id`, ":")[[1]][[2]]
issn_title[[issn]]

## End(Not run)
```

---

# orcid_person *Get personal data for a person*

---

## Description

Get personal data for a person

## Usage

```
orcid_person(orcid, details = FALSE, ...)
```

## Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| details | (logical). also get details. Default: FALSE |
| ... | Curl options passed on to crul::HttpClient() |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
res <- orcid_person(orcid = "0000-0002-9341-7985")
res$`0000-0002-9341-7985`
names(res$`0000-0002-9341-7985`)
res$`0000-0002-9341-7985`$`last-modified`
res$`0000-0002-9341-7985`$`keywords`
res$`0000-0002-9341-7985`$`biography`

## End(Not run)
```

---

`orcid_ping` *Check if ORCID API is up and running*

---

## Description

Check if ORCID API is up and running

## Usage

```
orcid_ping(...)
```

## Arguments

| | |
|---|---|
| `...` | Curl options passed on to `crul::HttpClient()` |

## Value

a text string

## Examples

```
## Not run:
orcid_ping()

## End(Not run)
```

## orcid_qualifications *Get qualifications for a person*

### Description

Get qualifications for a person

### Usage

```
orcid_qualifications(orcid, put_code = NULL,
  format = "application/json", summary = FALSE, ...)
```

### Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| summary | (logical) get peer review summary for a put code. Default: FALSE |
| ... | Curl options passed on to crul::HttpClient() |

### Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

### Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## orcid_researcher_urls *Get researcher urls for a person*

### Description

Get researcher urls for a person

### Usage

```
orcid_researcher_urls(orcid, put_code = NULL,
  format = "application/json", ...)
```

### Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| ... | Curl options passed on to `crul::HttpClient()` |

### Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

### Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

### Examples

```
## Not run:
# all data
res <- orcid_researcher_urls(orcid = "0000-0003-1444-9135")
res$`0000-0003-1444-9135`
names(res$`0000-0003-1444-9135`)
res$`0000-0003-1444-9135`$`researcher-url`

# individual ones
orcid_researcher_urls("0000-0003-1444-9135", 304093)
orcid_researcher_urls("0000-0003-1444-9135", c(332241, 304093))

# formats
orcid_researcher_urls("0000-0003-1444-9135", 304093,
  format = "application/xml")

## End(Not run)
```

---

## orcid_research_resources

*Get research resources for a person*

---

### Description

Get research resources for a person

### Usage

```
orcid_research_resources(orcid, put_code = NULL,
  format = "application/json", summary = FALSE, ...)
```

### Arguments

| | |
|---|---|
| orcid | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| put_code | (character/integer) one or more put codes. up to 50. optional |
| format | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| summary | (logical) get education summary for a put code. Default: FALSE |
| ... | Curl options passed on to crul::HttpClient() |

### Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

### Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

### Examples

```
## Not run:
res <- orcid_research_resources(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
res$`0000-0002-1642-628X`$`last-modified-date`
res$`0000-0002-1642-628X`$`group`
res$`0000-0002-1642-628X`$path

## End(Not run)
```

---

## orcid_search *Orcid search - more user friendly than* orcid()

---

### Description

Orcid search - more user friendly than orcid()

## Usage

```
orcid_search(given_name = NULL, family_name = NULL, past_inst = NULL,
  current_prim_inst = NULL, current_inst = NULL, credit_name = NULL,
  other_name = NULL, email = NULL, digital_object_ids = NULL,
  work_title = NULL, grant_number = NULL, patent_number = NULL,
  keywords = NULL, text = NULL, rows = 10, start = NULL, ...)
```

## Arguments

| | |
|---|---|
| `given_name` | (character) given name |
| `family_name` | (character) family name |
| `past_inst` | (character) past institution |
| `current_prim_inst` | (character) current primary institution |
| `current_inst` | (character) current institution |
| `credit_name` | (character) credit name |
| `other_name` | (character) other name |
| `email` | (character) email |
| `digital_object_ids` | (character) digital object ids |
| `work_title` | (character) work title |
| `grant_number` | (character) grant number |
| `patent_number` | (character) patent number |
| `keywords` | (character) keywords to search |
| `text` | (character) text to search |
| `rows` | (integer) number of records to return |
| `start` | (integer) record number to start at |
| `...` | curl options passed on to crul::HttpClient |

## Details

The goal of this function is to make a human friendly way to search ORCID.

Thus, internally we map the parameters given to this function to the actual parameters that ORCID wants that are not so human friendly.

## Value

a `data.frame` with three columns:

- first: given name
- last: family name
- orcid: ORCID identifier

### How parameters are combined

We combine multiple parameters with `AND`, such that e.g., `given_name="Jane"` and `family_name="Doe"` gets passed to ORCID as `given-names:Jane AND family-name:Doe`

### See Also

`orcid()`

### Examples

```
## Not run:
orcid_search(given_name = "carl", family_name = "boettiger")
orcid_search(given_name = "carl")

orcid_search(given_name = "carl", rows = 2)

orcid_search(given_name = "carl", verbose = TRUE)

## End(Not run)
```

---

## orcid_services *Get services*

---

### Description

Get services

### Usage

```
orcid_services(orcid, put_code = NULL, format = "application/json",
  summary = FALSE, ...)
```

### Arguments

| | |
|---|---|
| `orcid` | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| `put_code` | (character/integer) one or more put codes. up to 50. optional |
| `format` | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| `summary` | (logical) get education summary for a put code. Default: `FALSE` |
| `...` | Curl options passed on to `crul::HttpClient()` |

### Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
res <- orcid_services(orcid = "0000-0002-1642-628X")
res$`0000-0002-1642-628X`
res$`0000-0002-1642-628X`$`last-modified-date`
res$`0000-0002-1642-628X`$`affiliation-group`
res$`0000-0002-1642-628X`$path

## End(Not run)
```

---

`orcid_works` *Get works for a person*

---

## Description

Get works for a person

## Usage

```
orcid_works(orcid, put_code = NULL, format = "application/json", ...)
```

## Arguments

| | |
|---|---|
| `orcid` | (character) Orcid identifier(s), of the form XXXX-XXXX-XXXX-XXXX. required. |
| `put_code` | (character/integer) one or more put codes. up to 50. optional |
| `format` | (character) Name of the content-type format. One of "application/vnd.orcid+xml; qs=5", "application/orcid+xml; qs=3", "application/xml", "application/vnd.orcid+json; qs=4", "application/orcid+json; qs=2", "application/json" "application/vnd.citationstyles.csl+json". optional |
| `...` | Curl options passed on to `crul::HttpClient()` |

## Details

This function is vectorized, so you can pass in many ORCID's, and there's an element returned for each ORCID you put in.

## Value

A list of results for each Orcid ID passed in, with each element named by the Orcid ID

## Examples

```
## Not run:
# get all works
res <- orcid_works(orcid = "0000-0002-9341-7985")
res$`0000-0002-9341-7985`
res$`0000-0002-9341-7985`$works
res$`0000-0002-9341-7985`$works$type
str(res$`0000-0002-9341-7985`)

# get individual works
a <- orcid_works(orcid = "0000-0002-9341-7985", put_code = 5011717)
a$`0000-0002-9341-7985`
a$`0000-0002-9341-7985`$works
b <- orcid_works(orcid = "0000-0002-9341-7985",
   put_code = c(5011717, 15536016))
b$`0000-0002-9341-7985`

# change formats
orcid_works("0000-0002-9341-7985", 5011717, "application/json")
orcid_works("0000-0002-9341-7985", 5011717, "application/xml")
orcid_works("0000-0002-9341-7985", 5011717,
  "application/vnd.orcid+xml; qs=5")
orcid_works("0000-0002-9341-7985", 5011717,
  "application/vnd.citationstyles.csl+json")

# get citations
id <- "0000-0001-7678-8656"
x <- orcid_works(id)
wks <- orcid_works(id, put_code = x[[1]]$works$`put-code`)
wks[[1]]$works$`work.citation.citation-value`

## or send many put codes at once, will be split into chunks of 50 each
id <- "0000-0001-6758-5101"
z <- orcid_works(id)
pcodes <- z[[1]]$works$`put-code`
length(pcodes)
res <- orcid_works(orcid = id, put_code = pcodes)
head(res$`0000-0001-6758-5101`$works)

## End(Not run)
```

---

`rorcid-defunct` *Defunct functions in rorcid*

---

## Description

- `summary.or_cid()`: Function is gone. Deemed not really that useful, and hard to maintain given other changes in the package.

works *Get works data*

## Description

Get works data

## Usage

```
works(x)
```

## Arguments

| | |
|---|---|
| x | Anything that can be coerced via `as.orcid()`, see `as.orcid()` for help |
| ... | curl options passed on to crul::HttpClient |

## Details

This function gets works using the function orcid_works and packages up the data in a data.frame for easier processing

## Value

A tibble (data.frame)

## Examples

```
## Not run:
out <- works(orcid_id("0000-0002-9341-7985"))
out
out$type
out$path

works( orcid_id("0000-0002-1642-628X") )
works( orcid_id("0000-0003-1444-9135") )
works( orcid_id("0000-0003-1419-2405") )

out <- orcid(query="keyword:ecology")
works(orcid_id(out$`orcid-identifier.path`[7]))
works(orcid_id(out$`orcid-identifier.path`[8]))
works(orcid_id(out$`orcid-identifier.path`[9]))
works(orcid_id(out$`orcid-identifier.path`[10]))

## End(Not run)
```

# Index