# Package 'GenomicDataCommons'

April 11, 2018

**Type** Package

**Title** NIH / NCI Genomic Data Commons Access

**Description** Programmatically access the NIH / NCI Genomic Data Commons RESTful service.

**Version** 1.2.0

**Date** 2017-08-29

**License** Artistic-2.0

**Depends** R (>= 3.4.0), magrittr

**Imports** stats, httr, xml2, jsonlite, utils, lazyeval, readr, data.table, GenomicRanges, IRanges

**Suggests** BiocStyle, knitr, rmarkdown, DT, testthat, listviewer, ggplot2, GenomicAlignments, Rsamtools

**biocViews** DataImport, Sequencing

**URL** https://bioconductor.org/packages/GenomicDataCommons, http://github.com/Bioconductor/GenomicDataCommons

**BugReports** https://github.com/Bioconductor/GenomicDataCommons/issues/new

**VignetteBuilder** knitr

**RoxygenNote** 6.0.1

**NeedsCompilation** no

**Author** Martin Morgan [aut], Davis Sean [aut, cre]

**Maintainer** Davis Sean <sdavis2@mail.nih.gov>

## R topics documented:

---

`aggregations` *aggregations*

---

## Description

aggregations

## Usage

```
aggregations(x)

## S3 method for class 'GDCQuery'
aggregations(x)

## S3 method for class 'GDCResponse'
aggregations(x)
```

## Arguments

| | |
|---|---|
| x | a `GDCQuery` object |

## Value

a `list` of `data.frame` with one member for each requested facet. The data frames each have two columns, key and doc_count.

### Methods (by class)

- GDCQuery:
- GDCResponse:

### Examples

```
library(magrittr)
# Number of each file type
res = files() %>% facet(c('type','data_type')) %>% aggregations()
res$type
```

---

## as.data.frame.GDCResults

*Convert GDC results to data.frame*

---

### Description

GDC results are typically returned as an R list structure. This method converts that R list structure to a data.frame. Some columns in the resulting data.frame may remain lists, but there will be one list element for each row though that list element may contain multiple values.

### Usage

```
as.data.frame.GDCResults(x, row.names, optional, ...)
```

### Arguments

| | |
|---|---|
| x | a GDCResults object to be converted to a data.frame. |
| row.names | not used here; row.names are calculated from the data to maintain data integrity |
| optional | not used here; just for matching method call for as.data.frame |
| ... | not used here; just for matching method call for as.data.frame |

### Value

a data.frame, potentially with list columns still present.

### Note

The data.frame that is returned is the cartesian product of sub-data.frames that come from the GDC. This may lead to more than one row per entity id. It is up to the user to determine how to make rows unique per entity, if that is desired.

### Examples

```
expands = c("diagnoses","diagnoses.treatments","annotations",
            "demographic","exposures")
head(cases() %>% expand(expands) %>% results() %>% as.data.frame())
```

## available_expand *Return valid values for "expand"*

### Description

The GDC allows a shorthand for specifying groups of fields to be returned by the metadata queries. These can be specified in a `select` method call to easily supply groups of fields.

### Usage

```
available_expand(entity)

## S3 method for class 'character'
available_expand(entity)

## S3 method for class 'GDCQuery'
available_expand(entity)
```

### Arguments

| | |
|---|---|
| `entity` | Either a `GDCQuery` object or a character(1) specifying a GDC entity ('cases', 'files', 'annotations', 'projects') |

### Value

A character vector

### See Also

See https://docs.gdc.cancer.gov/API/Users_Guide/Search_and_Retrieval/#expand for details

### Examples

```
head(available_expand('files'))
```

## available_fields *S3 Generic to return all GDC fields*

### Description

S3 Generic to return all GDC fields

### Usage

```
available_fields(x)

## S3 method for class 'GDCQuery'
available_fields(x)

## S3 method for class 'character'
available_fields(x)
```

### Arguments

| | |
|---|---|
| x | A character(1) string ('cases','files','projects', 'annotations') or an subclass of GDCQuery. |

### Value

a character vector of the default fields

### Methods (by class)

- GDCQuery: GDCQuery method
- character: character method

### Examples

```
available_fields('projects')
projQuery = query('projects')
available_fields(projQuery)
```

---

## available_values *Find common values for a GDC field*

---

### Description

Find common values for a GDC field

### Usage

```
available_values(entity, field, legacy = FALSE)
```

### Arguments

| | |
|---|---|
| entity | character(1), a GDC entity ("cases", "files", "annotations", "projects") |
| field | character(1), a field that is present in the entity record |
| legacy | logical(1), use the legacy endpoint or not. |

### Value

character vector of the top 100 (or fewer) most frequent values for a the given field

## Examples

```
available_values('files','cases.project.project_id')[1:5]
```

---

`count` *provide count of records in a* `GDCQuery`

---

## Description

provide count of records in a `GDCQuery`

## Usage

```
count(x, ...)

## S3 method for class 'GDCQuery'
count(x, ...)

## S3 method for class 'GDCResponse'
count(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | a `GDCQuery` object |
| `...` | passed to httr (good for passing config info, etc.) |

## Value

integer(1) representing the count of records that will be returned by the current query

## Methods (by class)

- `GDCQuery`:
- `GDCResponse`:

## Examples

```
# total number of projects
projects() %>% count()

# total number of cases
cases() %>% count()
```

## default_fields *S3 Generic to return default GDC fields*

### Description

S3 Generic to return default GDC fields

### Usage

```
default_fields(x)

## S3 method for class 'character'
default_fields(x)

## S3 method for class 'GDCQuery'
default_fields(x)
```

### Arguments

| | |
|---|---|
| x | A character string ('cases','files','projects', 'annotations') or an subclass of GDCQuery. |

### Value

a character vector of the default fields

### Methods (by class)

- character: character method
- GDCQuery: GDCQuery method

### Examples

```
default_fields('projects')
projQuery = query('projects')
default_fields(projQuery)
```

## entity_name *Get the entity name from a GDCQuery object*

### Description

An "entity" is simply one of the four medata endpoints.

- cases
- projects
- files
- annotations

All GDCQuery objects will have an entity name. This S3 method is simply a utility accessor for those names.

### Usage

```
entity_name(x)

## S3 method for class 'GDCQuery'
entity_name(x)

## S3 method for class 'GDCResults'
entity_name(x)
```

### Arguments

| | |
|---|---|
| x | a GDCQuery object |

### Value

character(1) name of an associated entity; one of "cases", "files", "projects", "annotations".

### Examples

```
qcases = cases()
qprojects = projects()

entity_name(qcases)
entity_name(qprojects)
```

---

## expand *Set the* expand *parameter*

---

### Description

S3 generic to set GDCQuery expand parameter

### Usage

```
expand(x, expand)

## S3 method for class 'GDCQuery'
expand(x, expand)
```

### Arguments

| | |
|---|---|
| x | the objects on which to set fields |
| expand | a character vector specifying the fields |

### Value

A GDCQuery object, with the expand member altered.

### Methods (by class)

- GDCQuery: set expand fields on a GDCQuery object

## Examples

```
gProj = projects()
gProj$fields
head(available_fields(gProj))
default_fields(gProj)

gProj %>%
  select(default_fields(gProj)[1:2]) %>%
  response() %>%
  str(max_level=2)
```

---

`facet` *Set facets for a* `GDCQuery`

---

## Description

Set facets for a `GDCQuery`

Get facets for a `GDCQuery`

## Usage

```
facet(x, facets)

get_facets(x)

## S3 method for class 'GDCQuery'
get_facets(x)
```

## Arguments

| | |
|---|---|
| `x` | a `GDCQuery` object |
| `facets` | a character vector of fields that will be used for forming aggregations (facets). Default is to set facets for all default fields. See `default_fields` for details |

## Value

returns a `GDCQuery` object, with facets field updated.

## Examples

```
# create a new GDCQuery against the projects endpoint
gProj = projects()

# default facets are NULL
get_facets(gProj)

# set facets and save result
gProjFacet = facet(gProj)

# check facets
get_facets(gProjFacet)
```

```
# and get a response, noting that
# the aggregations list member contains
# tibbles for each facet
str(response(gProjFacet,size=2),max.level=2)
```

---

`filtering` *Manipulating GDCQuery filters*

---

## Description

Manipulating GDCQuery filters

The `filter` is simply a safe accessor for the filter element in `GDCQuery` objects.

The `get_filter` is simply a safe accessor for the filter element in `GDCQuery` objects.

## Usage

```
filter(x, expr)

## S3 method for class 'GDCQuery'
filter(x, expr)

get_filter(x)

## S3 method for class 'GDCQuery'
get_filter(x)
```

## Arguments

| | |
|---|---|
| `x` | the object on which to set the filter list member |
| `expr` | a filter expression in the form of the right hand side of a formula, where bare names (without quotes) are allowed if they are available fields associated with the GDCQuery object, `x` |

## Value

A `GDCQuery` object with the filter field replaced by specified filter expression

## Examples

```
# make a GDCQuery object to start
#
# Projects
#
pQuery = projects()

# check for the default fields
# so that we can use one of them to build a filter
default_fields(pQuery)
pQuery = filter(pQuery,~ project_id == 'TCGA-LUAC')
get_filter(pQuery)
```

```
#
# Files
#
fQuery = files()
default_fields(fQuery)

fQuery = filter(fQuery,~ data_format == 'VCF')
get_filter(fQuery)

fQuery = filter(fQuery,~ data_format == 'VCF' & experimental_strategy == 'WXS' & type == 'simple_somatic_muta

# Use str() to get a cleaner picture
str(get_filter(fQuery))
```

---

gdcdata *Download GDC files*

---

## Description

Download one or more files from GDC. Files are downloaded using the UUID and renamed to the file name on the remote system. By default, neither the uuid nor the file name on the remote system can exist.

## Usage

```
gdcdata(uuids, destination_dir = tempfile(), overwrite = FALSE,
  progress = interactive(), token = NULL)
```

## Arguments

| | |
|---|---|
| uuids | character() of GDC file UUIDs. |
| destination_dir | character(1) file path to a directory for downloading files. |
| overwrite | logical(1) default FALSE indicating whether existing files with identical name should be over-written. |
| progress | logical(1) default TRUE in interactive sessions, FALSE otherwise indicating whether a progress par should be produced for each file download. |
| token | (optional) character(1) security token allowing access to restricted data. See https://gdc-docs.nci.nih.gov/API/Users_Guide/Authentication_and_Authorization/. |

## Details

This function is appropriate for one or several files; for large downloads use manifest to create a manifest for and the GDC Data Transfer Tool.

## Value

a named vector with file uuids as the names and paths as the value

## See Also

manifest for downloading large data.

## Examples

```
uuids <- c("e3228020-1c54-4521-9182-1ea14c5dc0f7",
           "18e1e38e-0f0a-4a0e-918f-08e6201ea140")
(files <- gdcdata(uuids, overwrite=TRUE))
setNames(file.size(files), names(files))
```

---

# gdc_client *return gdc-client executable path*

---

## Description

This function is a convenience function to find and return the path to the GDC Data Transfer Tool executable assumed to be named 'gdc-client'. The assumption is that the appropriate version of the GDC Data Transfer Tool is a separate download available from https://gdc.cancer.gov/access-data/gdc-data-transfer-tool and as a backup from https://github.com/NCI-GDC/gdc-client. The following locations are checked:

## Usage

```
gdc_client()
```

## Details

- Sys.which() to see if gdc-client is on the path
- The current working directory
- The file name specified in the environment variable GDC_CLIENT

## Value

character(1) the path to the gdc-client executable.

## Examples

```
# this cannot run without first
# downloading the GDC Data Transfer Tool
gdc_client = try(gdc_client(),silent=TRUE)
```

gdc_token *return a gdc token from file or environment*

## Description

The GDC requires an auth token for downloading data that are "controlled access". For example, BAM files for human datasets, germline variant calls, and SNP array raw data all are protected as "controlled access". For these files, a GDC access token is required. See the https://docs.gdc.cancer.gov/Data_Portal/User authentication-tokens. Note that this function simply returns a string value. It is possible to keep the GDC token in a variable in R or to pass a string directly to the appropriate parameter. This function is simply a convenience function for alternative approaches to get a token from an environment variable or a file.

## Usage

```
gdc_token()
```

## Details

This function will resolve locations of the GDC token in the following order:

- from the environment variable, `GDC_TOKEN`, expected to contain the token downloaded from the GDC as a string
- using `readLines` to read a file named in the environment variable, `GDC_TOKEN_FILE`
- using `readLines` to read from a file called `.gdc_token` in the user's home directory

If all of these fail, this function will return an error.

## Value

character(1) (invisibly, to protect against inadvertently printing) the GDC token.

## References

https://docs.gdc.cancer.gov/Data_Portal/Users_Guide/Authentication/#gdc-authentication-tokens

## Examples

```
# This will not run before a GDC token
# is in place.
token = try(gdc_token(),silent=TRUE)
```

GenomicDataCommons *ncigdc: A package for computating the notorious bar statistic.*

## Description

Cool package for interfacing with NCI GDC

## finding data

- query
- cases
- projects
- files
- annotations
- mapping

## downloading data

data

---

grep_fields *Find matching field names*

## Description

This utility function allows quick text-based search of available fields for using grep

## Usage

```
grep_fields(entity, pattern, ..., value = TRUE)
```

## Arguments

| | |
|---|---|
| entity | one of "files", "cases", "annotations", "projects" against which to gather available fields for matching |
| pattern | A regular expression that will be used in a call to grep |
| ... | passed on to grep |
| value | logical(1) whether to return values as opposed to indices (passed along to grep) |

## Value

character() vector of field names matching pattern

## Examples

```
grep_fields('files','analysis')
```

## ids *Get the ids associated with a GDC query or response*

### Description

The GDC assigns ids (in the form of uuids) to objects in its database. Those ids can be used for relationships, searching on the website, and as unique ids. All

### Usage

```
ids(x)

## S3 method for class 'GDCQuery'
ids(x)

## S3 method for class 'GDCResults'
ids(x)

## S3 method for class 'GDCResponse'
ids(x)
```

### Arguments

| | |
|---|---|
| x | A GDCQuery or GDCResponse object |

### Value

a character vector of all the entity ids

### Examples

```
# use with a GDC query, in this case for "cases"
ids(cases() %>% filter(~ project.project_id == "TCGA-CHOL"))
# also works for responses
ids(response(files()))
# and results
ids(results(cases()))
```

## id_field *get the name of the id field*

### Description

In many places in the GenomicDataCommons package, the entity ids are stored in a column or a vector with a specific name that corresponds to the field name at the GDC. The format is the entity name (singular) "_id". This generic simply returns that name from a given object.

### Usage

```
id_field(x)

## S3 method for class 'GDCQuery'
id_field(x)

## S3 method for class 'GDCResults'
id_field(x)
```

### Arguments

| | |
|---|---|
| x | An object representing the query or results of an entity from the GDC ("cases", "files", "annotations", "projects") |

### Value

character(1) such as "case_id", "file_id", etc.

### Methods (by class)

- `GDCQuery`: GDCQuery method
- `GDCResults`: GDCResults method

### Examples

```
id_field(cases())
```

---

## make_filter *Create NCI GDC filters for limiting GDC query results*

---

### Description

Searching the NCI GDC allows for complex filtering based on logical operations and simple comparisons. This function facilitates writing such filter expressions in R-like syntax with R code evaluation.

### Usage

```
make_filter(expr, available_fields)
```

### Arguments

| | |
|---|---|
| expr | a filter expression |
| available_fields | a character vector of the additional names that will be injected into the filter evaluation environment |

### Details

If used with available_fields, "bare" fields that are named in the available_fields character vector can be used in the filter expression without quotes.

### Value

a `list` that represents an R version of the JSON that will ultimately be used in an NCI GDC search or other query.

---

`manifest` *Prepare GDC manifest file for bulk download*

---

### Description

The `manifest` function/method creates a manifest of files to be downloaded using the GDC Data Transfer Tool. There are methods for creating manifest data frames from `GDCQuery` objects that contain file information ("cases" and "files" queries).

### Usage

```
manifest(x, from = 0, size = count(x), ...)

## S3 method for class 'gdc_files'
manifest(x, from = 0, size = count(x), ...)

## S3 method for class 'GDCfilesResponse'
manifest(x, from = 0, size = count(x), ...)

## S3 method for class 'GDCcasesResponse'
manifest(x, from = 0, size = count(x), ...)
```

### Arguments

| | |
|---|---|
| `x` | An `GDCQuery` object of subclass "gdc_files" or "gdc_cases". |
| `from` | Record number from which to start when returning the manifest. |
| `size` | The total number of records to return. Default will return the usually desirable full set of records. |
| `...` | passed to `PUT`. |

### Value

A `tibble`, also of type "gdc_manifest", with five columns:

- id
- filename
- md5
- size
- state

### Methods (by class)

- `gdc_files`:
- `GDCfilesResponse`:
- `GDCcasesResponse`:

## Examples

```
gFiles = files()
shortManifest = gFiles %>% manifest(size=10)
head(shortManifest,n=3)
```

---

`mapping` *Query GDC for available endpoint fields*

---

## Description

Query GDC for available endpoint fields

## Usage

```
mapping(endpoint)
```

## Arguments

| | |
|---|---|
| `endpoint` | character(1) corresponding to endpoints for which users may specify additional or alternative fields. Endpoints include “projects”, “cases”, “files”, and “annotations”. |

## Value

A data frame describing the field (field name), full (full data model name), type (data type), and four additional columns describing the "set" to which the fields belong–“default”, “expand”, “multi”, and “nested”.

## Examples

```
map <- mapping("projects")
head(map)
# get only the "default" fields
subset(map,defaults)
# And get just the text names of the "default" fields
subset(map,defaults)$field
```

---

`query` *Start a query of GDC metadata*

---

## Description

The basis for all functionality in this package starts with constructing a query in R. The GDCQuery object contains the filters, facets, and other parameters that define the returned results. A token is required for accessing certain datasets.

## Usage

```
query(entity, filters = NULL, facets = NULL, legacy = FALSE,
  expand = NULL, fields = default_fields(entity))

cases(...)

files(...)

projects(...)

annotations(...)
```

## Arguments

| | |
|---|---|
| entity | character vector of 'cases','files','annotations', or 'projects' |
| filters | a filter list, typically created using make_filter, or added to an existing GDCQuery object using filter. |
| facets | a character vector of |
| legacy | logical(1) whether to use the "legacy" archive or not. See https://docs.gdc.cancer.gov/Data_Portal/Users_Guide/Legacy_Archive/ and https://gdc-portal.nci.nih.gov/legacy-archive/search/f for details. |
| expand | a character vector of "expands" to include in returned data |
| fields | a character vector of fields to return |
| ... | passed through to query |

## Value

An S3 object, the GDCQuery object. This is a list with the following members.

- filters
- facets
- fields
- expand
- archive
- token

## Functions

- cases: convenience contructor for a GDCQuery for cases
- files: convenience contructor for a GDCQuery for cases
- projects: convenience contructor for a GDCQuery for cases
- annotations: convenience contructor for a GDCQuery for annotations

## Examples

```
qcases = query('cases')
# equivalent to:
qcases = cases()
```

## rbindlist2 *rbindlist that deals with* NULL *rows*

### Description

This is a simple function that removes null values from the input list before applying rbindlist.

### Usage

```
rbindlist2(x, ...)
```

### Arguments

| | |
|---|---|
| x | a list that is appropriate for input to rbindlist, but may include NULLs, which will be filtered. |
| ... | passed directly to rbindlist. |

### Value

a data.table,data.frame object.

### Examples

```
input = list(list(a=1,b=2,d='only this row'),
             NULL,
             list(a=3,b=4))
rbindlist2(input)
```

## readDNAcopy *Read DNAcopy results into GRanges object*

### Description

Read DNAcopy results into GRanges object

### Usage

```
readDNAcopy(fname, ...)
```

### Arguments

| | |
|---|---|
| fname | The path to a DNAcopy-like file. |
| ... | passed to read_tsv |

### Value

a GRanges object

## Examples

```
fname = system.file(package='GenomicDataCommons',
                    'extdata/dnacopy.tsv.gz')
dnac = readDNAcopy(fname)
class(dnac)
length(dnac)
```

---

`readHTSeqFile` *Read a single htseq-counts result file.*

---

## Description

The htseq package is used extensively to count reads relative to regions (see http://www-huber.embl.de/HTSeq/doc/counting.html). The output of htseq-count is a simple two-column table that includes features in column 1 and counts in column 2. This function simply reads in the data from one such file and assigns column names.

## Usage

```
readHTSeqFile(fname, samplename = "sample", ...)
```

## Arguments

| | |
|---|---|
| `fname` | character(1), the path of the htseq-count file. |
| `samplename` | character(1), the name of the sample. This will become the name of the second column on the resulting `data.frame`, making for easier merging if necessary. |
| `...` | passed to `read_tsv`) |

## Value

a two-column data frame

## Examples

```
fname = system.file(package='GenomicDataCommons',
                    'extdata/example.htseq.counts.gz')
dat = readHTSeqFile(fname)
head(dat)
```

response *Fetch GDCQuery metadata from GDC*

## Description

Fetch GDCQuery metadata from GDC

## Usage

```
response(x, ...)

## S3 method for class 'GDCQuery'
response(x, from = 0, size = 10, ...,
  response_handler = jsonlite::fromJSON)

response_all(x, ...)
```

## Arguments

| | |
|---|---|
| x | a GDCQuery object |
| ... | passed to httr (good for passing config info, etc.) |
| from | integer index from which to start returning data |
| size | number of records to return |
| response_handler | a function that processes JSON (as text) and returns an R object. Default is fromJSON. |

## Value

A GDCResponse object which is a list with the following members:

- results
- query
- aggregations
- pages

## Examples

```
# basic class stuff
gCases = cases()
resp = response(gCases)
class(resp)
names(resp)

# And results from query
resp$results[[1]]
```

`results` *results*

## Description

results

## Usage

```
results(x, ...)

## S3 method for class 'GDCQuery'
results(x, ...)

## S3 method for class 'GDCResponse'
results(x, ...)
```

## Arguments

| | |
|---|---|
| x | a `GDCQuery` object |
| ... | passed on to `response` |

## Value

A (typically nested) `list` of GDC records

## Methods (by class)

- `GDCQuery`:
- `GDCResponse`:

## Examples

```
qcases = cases() %>% results()
length(qcases)
```

`results_all` *results_all*

## Description

results_all

### Usage

```
results_all(x)

## S3 method for class 'GDCQuery'
results_all(x)

## S3 method for class 'GDCResponse'
results_all(x)
```

### Arguments

| | |
|---|---|
| x | a GDCQuery object |

### Value

A (typically nested) list of GDC records

### Methods (by class)

- GDCQuery:
- GDCResponse:

### Examples

```
# details of all available projects
projResults = projects() %>% results_all()
length(projResults)
count(projects())
```

---

## select *S3 generic to set GDCQuery fields*

---

### Description

S3 generic to set GDCQuery fields

### Usage

```
select(x, fields)

## S3 method for class 'GDCQuery'
select(x, fields)
```

### Arguments

| | |
|---|---|
| x | the objects on which to set fields |
| fields | a character vector specifying the fields |

### Value

A GDCQuery object, with the fields member altered.

### Methods (by class)

- GDCQuery: set fields on a GDCQuery object

### Examples

```
gProj = projects()
gProj$fields
head(available_fields(gProj))
default_fields(gProj)

gProj %>%
  select(default_fields(gProj)[1:2]) %>%
  response() %>%
  str(max_level=2)
```

---

## slicing *Query GDC for data slices*

---

### Description

This function returns a BAM file representing reads overlapping regions specified either as chromosomal regions or as gencode gene symbols.

### Usage

```
slicing(uuid, regions, symbols, destination = tempfile(), overwrite = FALSE,
  progress = interactive(), token = NULL, legacy = FALSE)
```

### Arguments

| | |
|---|---|
| uuid | character(1) identifying the BAM file resource |
| regions | character() vector describing chromosomal regions, e.g., c("chr1", "chr2:10000", "chr3:10000- (all of chromosome 1, chromosome 2 from position 10000 to the end, chromosome 3 from 10000 to 20000). |
| symbols | character() vector of gencode gene symbols, e.g., c("BRCA1", "PTEN") |
| destination | character(1) default tempfile() file path for BAM file slice |
| overwrite | logical(1) default FALSE can destination be overwritten? |
| progress | logical(1) default interactive() should a progress bar be used? |
| token | character(1) security token allowing access to restricted data. Almost all BAM data is restricted, so a token is usually required. See https://gdc-docs.nci.nih.gov/API/Users_Guide/Authentication_and_Authorization/. |
| legacy | logical(1) whether or not to use the "legacy" archive, containing older, non-harmonized data. |

### Value

character(1) destination to the downloaded BAM file

### Examples

```
slicing("df80679e-c4d3-487b-934c-fcc782e5d46e",
        regions="chr17:75000000-76000000",
        token=gdc_token())
```

---

## status *Query the GDC for current status*

---

### Description

Query the GDC for current status

### Usage

```
status(version = NULL)
```

### Arguments

| | |
|---|---|
| version | (optional) character(1) version of GDC |

### Value

List describing current status.

### Examples

```
status()
```

---

## transfer *Bulk data download*

---

### Description

The GDC maintains a special tool, https://docs.gdc.cancer.gov/Data_Transfer_Tool/Users_Guide/Getting_Started/, that enables high-performance, potentially parallel, and resumable downloads. The Data Transfer Tool is an external program that requires separate download.

`transfer_help()` queries the the command line GDC Data Transfer Tool, `gdc-client`, for available options to be used in the `transfer` command.

### Usage

```
transfer(manifest, destination_dir = tempfile(), args = character(),
  token = NULL, gdc_client = "gdc-client")

transfer_help(gdc_client = "gdc-client")
```

### Arguments

| | |
|---|---|
| manifest | character(1) file path to manifest created by `manifest()`. See `write_manifest` for a simple way to create a manifest file from a data.frame created with `manifest`. |
| destination_dir | The path into which to place the transfered files. |
| args | character() vector specifying command-line arguments to be passed to `gdc-client`. See `transfer_help` for possible values. The arguments `--manifest`, `--dir`, and `--token-file` are determined by `manifest`, `destination_dir`, and `token`, respectively, and should NOT be provided as elements of `args`. |
| token | character(1) containing security token allowing access to restricted data. See https://gdc-docs.nci.nih.gov/API/Users_Guide/Authentication_and_Authorization/. Note that the GDC transfer tool requires a file for data transfer. Therefore, this token will be written to a temporary file (with appropriate permissions set). |
| gdc_client | character(1) name or path to `gdc-client` executable. On Windows, use `/` or `\\` as the file separator. |

### Value

character(1) directory path to which the files were downloaded.

### Functions

- `transfer_help`:

### Examples

```
file_manifest = files() %>% filter(~ access == "open") %>% manifest(size=10)
manifest_file = tempfile()
write.table(file_manifest,file=manifest_file,col.names=TRUE,row.names=FALSE,quote=FALSE)
destination <- transfer(manifest_file)
dir(destination)
# and with authenication
destination <- transfer(manifest_file,token=gdc_token)
```

---

## write_manifest *write a manifest data.frame to disk*

---

### Description

The `manifest` method creates a data.frame that represents the data for a manifest file needed by the GDC Data Transfer Tool. While the file format is nothing special, this is a simple helper function to write a manifest data.frame to disk. It returns the path to which the file is written, so it can be used "in-line" in a call to `transfer`.

### Usage

```
write_manifest(manifest, destfile = tempfile())
```

## Arguments

| | |
|---|---|
| manifest | A data.frame with five columns, typically created by a call to manifest |
| destfile | The filename for saving the manifest. |

## Value

character(1) the destination file name.

## Examples

```
mf = files() %>% manifest(size=10)
write_manifest(mf)
```

# Index