Package 'HTAN'

November 20, 2026

Type Package

Title Query and Retrieve Human Tumor Atlas Network (HTAN) Open-Access Data

Version 0.99.1

Date 2026-10-14

Description A client for the open-access data of the Human Tumor Atlas Network (HTAN), Phase 1. File, biospecimen, clinical and provenance metadata are queried locally from a versioned parquet snapshot of the ISB-CGC HTAN BigQuery release tables, through DuckDB and dbplyr, with no Google account. Open-access files are downloaded from the CRDC General Commons (anonymous GA4GH DRS) or Synapse (personal access token), and common HTAN file formats are read into Bioconductor classes such as SingleCellExperiment, SpatialExperiment and MultiAssayExperiment.

License MIT + file LICENSE

URL https://github.com/seandavi/HTAN, https://seandavi.github.io/HTAN/

BugReports https://github.com/seandavi/HTAN/issues

Depends R (>= 4.6.0)

SystemRequirements Quarto command line tool (<https://github.com/quarto-dev/quarto-cli>).

Imports BiocFileCache, DBI, dbplyr, dplyr, duckdb, httr2, jsonlite, methods, MultiAssayExperiment, rlang, S4Vectors, SingleCellExperiment, SpatialExperiment, stats, SummarizedExperiment, tibble, tools, utils

Suggests anndataR, BiocIO, ggplot2, knitr, Matrix, quarto, rhdf5, scales, scater, scrapper, SeuratObject, TENxIO, tidyr, testthat (>= 3.0.0), VisiumIO, withr

biocViews DataImport, DataRepresentation, BiomedicalInformatics, SingleCell, Spatial, ThirdPartyClient, Transcriptomics

VignetteBuilder quarto

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/HTAN

git_branch devel

git_last_commit 9efae6b

git_last_commit_date 2026-10-20

Repository Bioconductor 3.24

Date/Publication 2026-11-19

Author Sean Davis [aut, cre] (ORCID: <https://orcid.org/0000-0002-8991-6458>), National Cancer Institute [fnd] (Grant U24CA289073)

Maintainer Sean Davis <seandavi@gmail.com>

Contents

HTAN-package 2
htan_con 3
htan_disconnect 4
htan_download 5
htan_experiment 6
htan_files 7
htan_lineage 8
htan_snapshot_info 9
htan_tables 10
read_htan 10

Index **12**

`HTAN-package` *HTAN: query and retrieve Human Tumor Atlas Network open-access data*

Description

The package has one seam: `htan_con()` opens a DuckDB connection over a versioned parquet snapshot of the ISB-CGC HTAN release tables. Every other function talks only to that connection, so the backing store (local fixture, HTTPS mirror, future BigQuery or Phase 2 adapter) can change without touching the user API.

Details

See `vignette("HTAN")` for a walk-through and `adr/` in the source repository for design decisions.

Development

Almost all of the code, tests and documentation were written by an AI coding agent, Claude Code (Anthropic), working from the maintainer's design and under the maintainer's direction and review. The maintainer is responsible for the package and its maintenance. Commits written with the agent carry a `Co-Authored-By:` trailer in the source repository. No third-party code was copied into the package.

Author(s)

Maintainer: Sean Davis <seandavi@gmail.com> (ORCID)

Authors:

- Sean Davis <seandavi@gmail.com> (ORCID)

Other contributors:

- National Cancer Institute (Grant U24CA289073) [funder]

See Also

Useful links:

- https://github.com/seandavi/HTAN
- https://seandavi.github.io/HTAN/
- Report bugs at https://github.com/seandavi/HTAN/issues

`htan_con` *Connect to an HTAN metadata snapshot*

Description

`htan_con()` opens an in-memory DuckDB connection over a versioned parquet snapshot of the ISB-CGC HTAN release tables. It reads the release's `manifest.json`, fetches each parquet file, checks its sha256 against the manifest, caches it in a BiocFileCache (for `http(s)://` snapshots), and creates one SQL `VIEW` per table. All other functions in the package take this connection, so changing where the snapshot lives never changes the API.

Usage

```
htan_con(
  release = "r7",
  base_url = getOption("HTAN.base_url", .default_base_url),
  cache = getOption("HTAN.cache", tools::R_user_dir("HTAN", "cache")),
  refresh = FALSE
)
```

Arguments

`release`	`character(1)`. The HTAN release, e.g. `"r7"`. Always explicit, so results are reproducible.
`base_url`	`character(1)`. Where the snapshot lives: an `https://` or `http://` URL, a local directory, or a `file://` URL. If it contains `{release}`, that is replaced by `release` (the Hugging Face layout, where the release is a tag); otherwise the release is a subdirectory, `<base_url>/<release>/manifest.json` (the layout `just snapshot` writes). Defaults to `getOption("HTAN.base_url")`, then the Hugging Face dataset.
`cache`	`character(1)`. BiocFileCache directory for downloaded parquet files. Defaults to `getOption("HTAN.cache")`, then `tools::R_user_dir("HTAN", "cache")`.
`refresh`	`logical(1)`. Open a new connection even if one is cached.

Details

By default the snapshot comes from the public Hugging Face dataset https://huggingface.co/datasets/seandavis/htan, one git tag per release (r7). Set options(HTAN.base_url = ...) to use a mirror or a local build.

The connection is cached for the session: calling htan_con() again with the same arguments returns the same connection.

Value

A duckdb_connection (a DBI connection).

See Also

htan_tables(), htan_tbl(), htan_files(), htan_disconnect()

Examples

```
fixture <- system.file("extdata", "snapshot", package = "HTAN")
con <- htan_con("r7", base_url = fixture)
htan_tables(con)
htan_disconnect()
```

---

htan_disconnect *Close cached HTAN connections*

Description

Close cached HTAN connections

Usage

```
htan_disconnect(con = NULL)
```

Arguments

con	A connection from htan_con(), or NULL (the default) to close every connection cached in this session.

Value

NULL, invisibly.

Examples

```
htan_disconnect()
```

htan_download *Download HTAN files*

Description

Fetches open-access HTAN files, routing each by its access tier from htan_files():

Usage

```
htan_download(
  files,
  dest = getOption("HTAN.download_dir"),
  overwrite = FALSE,
  max_bytes = getOption("HTAN.max_download_bytes", 2e+10),
  token = Sys.getenv("SYNAPSE_AUTH_TOKEN"),
  con = htan_con()
)
```

Arguments

files	The files to fetch: the result of htan_files() (lazy or collected), or a character vector of HTAN_Data_File_IDs.
dest	character(1) or NULL. A directory to copy the files to, as well as caching them. NULL (the default, unless option HTAN.download_dir is set) returns the paths in the cache.
overwrite	logical(1). Download again even if a verified copy is cached or present in dest.
max_bytes	numeric(1). Refuse to start if the files total more than this many bytes (Level 2 images are often several GB each).
token	character(1). Synapse personal access token, only needed for "open-synapse" files.
con	A connection from htan_con().

Details

- "open-gc": open Level 2 imaging in the CRDC General Commons, through anonymous GA4GH DRS (no account needed).
- "open-synapse": Level 3/4, auxiliary and other open files on Synapse, through the Synapse REST API. Needs a free Synapse account and a personal access token in SYNAPSE_AUTH_TOKEN.
- "controlled": Level 1/2 sequencing (dbGaP phs002371). Not supported; requesting one is an error.

Downloads are cached. Each file is kept in a BiocFileCache, keyed by its HTAN_Data_File_ID and md5, under the same relative path as HTAN's Filename (so 10x/Visium bundles stay together), and checked against the md5 in the metadata before it enters the cache. Asking for a file again is served from the cache after checking its md5 again; a cached copy that no longer matches is downloaded afresh. Nothing is written to the working directory.

The cache is the downloads directory inside the snapshot cache of htan_con() (option HTAN.cache, then tools::R_user_dir("HTAN", "cache")), or option HTAN.download_cache if set. It is a

BiocFileCache of its own, so it can be moved or cleared without touching the snapshot, e.g. with `BiocFileCache::removebfc()`.

With `dest`, each file is also copied to `<dest>/<Filename>`, and a copy already there with the right md5 is used as is.

Value

A tibble, one row per file: `HTAN_Data_File_ID`, `access`, `path` (in the cache, or in `dest` if given) and `status`: `"downloaded"` (fetched now), `"cached"` (served from the cache) or `"present"` (already in `dest`).

Examples

```
fixture <- system.file("extdata", "snapshot", package = "HTAN")
con <- htan_con("r7", base_url = fixture)
## a 250 KB open IMC image, fetched anonymously from General Commons
tiff <- htan_files(drs_uri == "dg.4DFC/6231a7d4-7e68-11ee-b51d-e7db3b9207b8",
    con = con)

dl <- htan_download(tiff, con = con)
dl$status
## a second call is served from the cache
htan_download(tiff, con = con)$status
```

`htan_experiment` *Assemble HTAN data into a MultiAssayExperiment*

Description

Builds a `MultiAssayExperiment` whose experiments are HTAN files and whose `colData` is one row per participant from the clinical tables (`clinical_tier1_demographics`, plus the first `clinical_tier1_diagnosis` row per participant). The `sampleMap` links each experiment column (cell, spot or sample) to its participant:

Usage

```
htan_experiment(x, participant_col = NULL, con = htan_con())
```

Arguments

`x`	A named list of `SummarizedExperiments` (for example from `read_htan()`) with `HTAN_Data_File_IDs` as names, or the result of `htan_download()`, whose files are read with `read_htan()`.
`participant_col`	`character(1)` or `NULL`. For multi-participant files, the `colData` column with each column's participant or biospecimen ID.
`con`	A connection from `htan_con()`.

Details

- a file derived from one participant (about 98 percent of r7 files) maps every column to that participant;
- for a multi-participant file, `participant_col` names the `colData` column holding each column's `HTAN_Participant_ID` (or biospecimen ID, from which the participant is taken).

This is a thin first version: it does not yet join biospecimen annotations or follow `id_provenance` below the participant.

Value

A `MultiAssayExperiment`.

Examples

```
fixture <- system.file("extdata", "snapshot", package = "HTAN")
con <- htan_con("r7", base_url = fixture)
f <- dplyr::pull(htan_files(level == 4, participants = "HTA12_251",
    con = con), HTAN_Data_File_ID)[1]
se <- SummarizedExperiment::SummarizedExperiment(
    list(counts = matrix(1:6, 2, dimnames = list(c("g1", "g2"),
        c("c1", "c2", "c3")))))
mae <- htan_experiment(stats::setNames(list(se), f), con = con)
MultiAssayExperiment::sampleMap(mae)
```

`htan_files` *Find HTAN files*

Description

One row per HTAN data file, combining every file-level metadata table in the snapshot (the tables with `HTAN_Data_File_ID`, `Filename`, `File_Format` and `Component` columns) with the participants and biospecimens from `id_provenance` and the CRDC General Commons DRS URI from `gc_drs_map`. The result is lazy: filter it with dplyr, then pass it to `htan_download()`.

Usage

```
htan_files(..., participants = NULL, con = htan_con())
```

Arguments

`...`	Filter conditions passed to `dplyr::filter()`, e.g. `assay == "ScRNA-seq", level == 4`.
`participants`	`character()` or `NULL`. Keep files derived from any of these `HTAN_Participant_IDs`, including multi-participant files.
`con`	A connection from `htan_con()`.

Details

Columns:

HTAN_Data_File_ID, Filename, File_Format, File_Size, md5, Component, entityId, HTAN_Center, Data_Releas From the file's metadata table. `File_Format` is coarse (`hdf5` covers `.h5ad` and 10x `.h5`; `RData` covers `.rds`), so see also `extension`.

table The snapshot table the row came from. A file listed in several tables appears once, from the most recently released one (in r7, 260 mass-spectrometry files are also in the legacy r2_v2 `lipidomics_metadata` and `metabolomics_metadata` tables).

assay, level `Component` split into the assay and its level (1-4; `NA` for auxiliary and other files).

extension Lower-case file extension from `Filename`, e.g. `.h5ad`, `.rds`, `.csv`, `.tsv.gz`.

HTAN_Participant_ID, HTAN_Assayed_Biospecimen_ID From `id_provenance`. A file derived from several participants (about 1.6 percent of files in r7) lists them comma-separated; use the `participants` argument, which handles that case, rather than `==`.

n_participants Number of distinct participants.

drs_uri CRDC General Commons DRS ID, if the file is there.

access Where the file can be fetched: `"open-gc"` (open Level 2 imaging, anonymous DRS), `"open-synapse"` (Level 3/4, auxiliary and other open files, Synapse with a token), or `"controlled"` (Level 1/2 sequencing; dbGaP phs002371, not supported).

Value

A lazy `tbl_duckdb_connection`.

Examples

```
fixture <- system.file("extdata", "snapshot", package = "HTAN")
con <- htan_con("r7", base_url = fixture)
htan_files(con = con)
htan_files(level == 4, extension == ".h5ad", con = con)
htan_files(participants = "HTA12_15", access == "open-gc", con = con)
```

---

htan_lineage *Trace the provenance of HTAN files*

Description

Walks `id_provenance`, where each row links a file to one parent file (`HTAN_Parent_Data_File_ID`) and to the biospecimens and participant it was derived from. `htan_lineage()` follows those links up (to the Level 1 files a processed file came from), down (to everything derived from a file), or both.

Usage

```
htan_lineage(
  ids,
  direction = c("ancestors", "descendants", "both"),
  max_depth = 20L,
  con = htan_con()
)
```

Arguments

ids	character(). HTAN_Data_File_IDs, or a table with an HTAN_Data_File_ID column such as the result of htan_files().
direction	One of "ancestors" (default), "descendants" or "both".
max_depth	integer(1). Stop after this many generations.
con	A connection from htan_con().

Value

A lazy tbl_duckdb_connection with one row per (query file, related file): HTAN_Data_File_ID (the query), related_file_id, depth (0 for the file itself; negative for ancestors, positive for descendants), and the related file's Component, Filename, entityId, HTAN_Participant_ID, HTAN_Assayed_Biospecime and HTAN_Originating_Biospecimen_ID (comma-separated when several).

Examples

```
fixture <- system.file("extdata", "snapshot", package = "HTAN")
con <- htan_con("r7", base_url = fixture)
l4 <- htan_files(assay == "ScRNA-seq", level == 4, con = con)
htan_lineage(l4, con = con)
```

---

htan_snapshot_info *Snapshot metadata recorded in a connection*

Description

Snapshot metadata recorded in a connection

Usage

```
htan_snapshot_info(con = htan_con())
```

Arguments

con	A connection from htan_con().

Value

A named list of the manifest's scalar fields (release, phase, build date, ISB-CGC source, ...), plus location, where this release was read from.

Examples

```
fixture <- system.file("extdata", "snapshot", package = "HTAN")
htan_snapshot_info(htan_con("r7", base_url = fixture))
```

htan_tables *List and open HTAN snapshot tables*

Description

`htan_tables()` lists the tables in the snapshot with their category, row count and ISB-CGC source. `htan_tbl()` opens one as a lazy dbplyr table, so `dplyr::filter()`, `dplyr::select()` and joins run in DuckDB until you `dplyr::collect()`. `htan_columns()` describes columns, using the BigQuery column descriptions recorded in the manifest.

Usage

```
htan_tables(con = htan_con())

htan_tbl(name, con = htan_con())

htan_columns(table = NULL, con = htan_con())
```

Arguments

con	A connection from `htan_con()`.
name	`character(1)`. A table name from `htan_tables()`, e.g. `"biospecimen"` or `"scRNAseq_level4_metadata"`.
table	`character()` or `NULL`. Restrict to these tables.

Value

`htan_tables()` and `htan_columns()`: a tibble. `htan_tbl()`: a lazy `tbl_duckdb_connection`.

Examples

```
fixture <- system.file("extdata", "snapshot", package = "HTAN")
con <- htan_con("r7", base_url = fixture)
htan_tables(con)
htan_tbl("clinical_tier1_demographics", con)
htan_columns("gc_drs_map", con)
```

read_htan *Read an HTAN file into a Bioconductor object*

Description

Dispatches on the file extension (HTAN's `File_Format` is too coarse: `hdf5` covers `.h5ad` and 10x `.h5`, `RData` covers `.rds`):

Usage

```
read_htan(x, coords = NULL, ...)
```

Arguments

x	A file path, a Space Ranger outs/ directory, or the result of htan_download() (in which case every row's path is read).
coords	character(2) or NULL. For cell tables, the x and y centroid columns. NULL tries common names (X_centroid/ Y_centroid from MCMICRO, x/y, centroid_x/centroid_y, and others).
...	Passed to the underlying reader.

Details

.h5ad anndataR::read_h5ad() to a SingleCellExperiment; a spatial embedding becomes spatialCoords() of a SpatialExperiment. Files in the pre-0.8 AnnData layout, which anndataR rejects and many HTAN centers wrote, are read with rhdf5 instead: X, layers, obs, var and numeric obsm arrays, but not uns. Duplicated obs or var names are made unique with make.unique(), with a message.

.rds, .rda, .RData Read, then: SingleCellExperiment and other SummarizedExperiments pass through; a Seurat object is converted with SeuratObject (counts, data, metadata, reductions, and tissue coordinates when present). Anything else is returned as is, with a message.

.h5 **(10x)** TENxIO::TENxH5() to a SingleCellExperiment.

a Space Ranger outs/ **directory** VisiumIO::TENxVisium() to a SpatialExperiment.

.csv, .tsv, .txt **(optionally** .gz**)** Imaging Level 4 cell tables become a SpatialExperiment (cells as columns, marker intensities as the intensities assay, centroids as spatialCoords(), other per-cell columns as colData). A table without recognisable coordinates is returned as a tibble.

.tif, .tiff, .svs Images are not read; the path is returned with a message (see RBioFormats or cytomapper).

Level 4 objects are heterogeneous across HTAN centers. Readers are best-effort: check the result.

Value

A SingleCellExperiment, SpatialExperiment, other SummarizedExperiment, tibble or path, depending on the file. For a multi-row htan_download() result, a list named by HTAN_Data_File_ID.

Examples

```
cells <- data.frame(CellID = 1:3, CD3 = c(1, 5, 2), CD8 = c(0, 3, 1),
    X_centroid = c(10, 20, 30), Y_centroid = c(5, 15, 25), Area = 40:42)
path <- tempfile(fileext = ".csv")
write.csv(cells, path, row.names = FALSE)
read_htan(path)

## A tiny Visium file in the pre-0.8 AnnData layout
if (requireNamespace("rhdf5", quietly = TRUE)) {
    read_htan(system.file("extdata", "h5ad", "legacy_visium.h5ad",
        package = "HTAN"))
}
```

Index

∗ **internal**
HTAN-package, 2

dplyr::filter(), *7*

HTAN *(*HTAN-package*)*, 2
HTAN-package, 2
htan_columns *(*htan_tables*)*, 10
htan_con, 3
htan_con(), *4–7*, *9*, *10*
htan_disconnect, 4
htan_disconnect(), *4*
htan_download, 5
htan_download(), *6*, *7*, *11*
htan_experiment, 6
htan_files, 7
htan_files(), *4*, *5*, *9*
htan_lineage, 8
htan_snapshot_info, 9
htan_tables, 10
htan_tables(), *4*
htan_tbl *(*htan_tables*)*, 10
htan_tbl(), *4*

make.unique(), *11*

read_htan, 10
read_htan(), *6*