Package: allofus (via r-universe)

August 22, 2024

Type Package

Title Interface for 'All of Us' Researcher Workbench

Version 1.2.0

Description Streamline use of the 'All of Us' Researcher Workbench (<https://www.researchallofus.org/data-tools/workbench/>)with tools to extract and manipulate data from the 'All of Us' database. Increase interoperability with the Observational Health Data Science and Informatics ('OHDSI') tool stack by decreasing reliance of 'All of Us' tools and allowing for cohort creation via 'Atlas'. Improve reproducible and transparent research using 'All of Us'.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports cli, tidyr, magrittr, dplyr (>= 1.1.4), glue, bigrquery (>= 1.5.1), purrr, stats, utils, dbplyr (>= 2.5.0), sessioninfo, rlang, stringr, DBI, lifecycle, bit64

Suggests knitr, rmarkdown, testthat (>= 3.0.0), kableExtra, DT, googlesheets4, tibble, forcats, gh, SqlRender (>= 1.6.0)

RoxygenNote 7.3.2

URL https://roux-ohdsi.github.io/allofus/, https://github.com/roux-ohdsi/allofus

BugReports https://github.com/roux-ohdsi/allofus/issues

VignetteBuilder knitr

Depends R (>= 2.10)

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

Repository https://roux-ohdsi.r-universe.dev

RemoteUrl https://github.com/roux-ohdsi/allofus

RemoteRef HEAD

RemoteSha eeb0637dda57e160983666c04637826009a93092

Contents

`aou_atlas_cohort` *Retrieve a cohort from ATLAS for use in All of Us*

Description

Retrieves a cohort definition from ATLAS and generates the cohort in All of Us. Observation periods are first generated for each subject using the `aou_observation_period()` function.The resulting cohort is a table with the cohort start and end dates for each person_id.

Usage

```
aou_atlas_cohort(
  cohort_definition,
  cohort_sql,
  debug = FALSE,
  collect = FALSE,
  ...,
  con = getOption("aou.default.con")
)
```

Arguments

`cohort_definition`	A cohort definition generated using `getCohortDefinition() from ROhdsiWebApi`
`cohort_sql`	The cohort_sql generated using `getCohortSql() from ROhdsiWebApi`
`debug`	Print the query to the console; useful for debugging.
`collect`	Whether to bring the resulting table into local memory (`collect = TRUE`) as a dataframe or leave as a reference to a database table (for continued analysis using, e.g., `dbplyr`). Defaults to `FALSE`.
`...`	Further arguments passed along to `collect()` if `collect = TRUE`
`con`	Connection to the allofus SQL database. Defaults to getOption("aou.default.con"), which is set automatically if you use `aou_connect()`

Details

The function is based on a similar function in `https://github.com/cmayer2/r4aou` with some tweaks to generate the appropriate observation periods and incorporate other package functions. Please see the online vignette for additional details.

Value

A dataframe if `collect = TRUE`; a reference to a remote database table if not. The SQL query used to generate the cohort is stored as an attribute.

Examples

```
# generate a simple stroke cohort
# see https://atlas-demo.ohdsi.org/#/cohortdefinition/1788061
# If this cohort is not available, you can create one, or choose one already made.
# aou_cohort_example contains the results of
# cd <- ROhdsiWebApi::getCohortDefinition(1788061, "https://atlas-demo.ohdsi.org/WebAPI")
# for some cohorts, you must use the argument generateStats = FALSE or the cohort (its stats)
# can't be generated on All of Us
# cd_sql <- ROhdsiWebApi::getCohortSql(cd, "https://atlas-demo.ohdsi.org/WebAPI",
#                                                   generateStats = FALSE)

cohort <- aou_atlas_cohort(
  cohort_definition = aou_cohort_example$cd,
  cohort_sql = aou_cohort_example$cd_sql
)

# print query that was executed
cat(attr(cohort, "query"))
```

`aou_bucket_to_workspace` *Move files from a bucket to your workspace*

Description

Retrieves a file from the workspace bucket and moves it into the current persistent disk where it can be read into R, e.g., using a function like read.csv().

Usage

```
aou_bucket_to_workspace(
  file,
  directory = FALSE,
  bucket = getOption("aou.default.bucket")
)
```

Arguments

`file`	The name of a file in your bucket, a vector of multiple files, a directory, or a file pattern (e.g. ".csv").
`directory`	Whether `file` refers to an entire directory you want to move.
`bucket`	Bucket to retrieve file from. Defaults to `getOption("aou.default.bucket")`, which is `Sys.getenv('WORKSPACE_BUCKET')` unless specified otherwise.

Details

This function retrieves a file from your bucket and moves it into your workspace where it can be read into R, e.g., using a function like `write.csv()`. See `https://cloud.google.com/storage/docs/gsutil/commands/cp` for details on the underlying function.

Value

Nothing

Examples

```
# save a file to the bucket
tmp <- tempdir()
write.csv(data.frame(x = 1), file.path(tmp, "testdata.csv"))
aou_workspace_to_bucket(file.path(tmp, "testdata.csv"))
# read the file back into the workspace
aou_bucket_to_workspace("testdata.csv")
# read in to your local environment
read.csv("testdata.csv")
```

aou_codebook *All of Us Modified Codebook*

Description

A data frame with rows from the publicly available All of Us Survey Codebook mapped to the All of Us PPI Vocabulary available on Athena. A small number of rows did not match between the codebook and the Athena PPI Vocabulary.

Usage

```
aou_codebook
```

Format

`aou_codebook` A data frame with 702 rows and 11 columns:

concept_code chr; Concept code from AOU codebook

concept_id int; mapped concept_id from PPI vocabulary

concept_name chr; Formatted text name of concept

concept_class_id chr; type of survey item - question or answer

form_name int; name of survey

field_type chr; type of question (radio, text, checkbox etc.)

field_label chr; The actual text of the question or answer

choices int; choices for question if radio or checkbox

standard_concept chr; Whether concept_id is a standard omop concept

valid_start_Date chr; start date for concept

valid_end_Date int; end date for concept

link chr; link to survey pdf

Details

Questions relating to specific conditions are not included as part of this table. They are instead available in the `aou_health_history` table.

- All of Us codebook
- Code to generate table

aou_collect *Collect a tbl object and convert integer64 columns to double*

Description

If you connect to the All of Us database via `aou_connect()`, integer columns will be converted to the int64 class, which can represent 64-bit integers. This is safer than keeping as R's default integer class, because some of the values of the ID columns in All of Us are larger than R can handle as integers. However, this can make working with the local table more difficult in RStudio as a vector of values will not match the int64 class. This is not a problem in Jupyter notebooks, meaning that code that works on one platform may not work on another. A safe practice is to use `aou_collect()`, which works just like `dplyr::collect()` except that any integer values are converted to doubles. If this is not what you want, set `convert_int64 = FALSE`.

Usage

```
aou_collect(data, convert_int64 = TRUE, ...)
```

Arguments

`data`	A reference to a remote database table (or unexecuted query)
`convert_int64`	Do you want to convert integer values to doubles? Defaults to `TRUE`
`...`	Other arguments passed to dplyr::collect()

Details

[Experimental]

Value

a local dataframe

Examples

```
# returns 2 rows, as expected
dplyr::tbl(con, "concept") %>%
  dplyr::filter(concept_id %in% c(1112807, 4167538)) %>%
  aou_collect() %>%
  dplyr::filter(concept_id %in% c(1112807, 4167538))

default_collect <- tbl(con, "concept") %>%
  dplyr::filter(concept_id %in% c(1112807, 4167538)) %>%
  dplyr::collect()
# returns 2 rows in Jupyter and 0 in RStudio
dplyr::filter(default_collect, concept_id %in% c(1112807, 4167538))
```

aou_compute — *Compute a dplyr tbl SQL query into a temp table*

Description

Computes a temporary table from a dplyr chain that returns an SQL query (e.g., tbl(con, table)) and returns the name of the temporary table. May be useful to create intermediate tables to reduce long queries. The temporary table will only exist for the current session and will nee to be created again a new session.

Usage

```
aou_compute(data, ..., con = getOption("aou.default.con"))
```

Arguments

`data`	A reference to an unexecuted remote query (e.g., the result of a `tbl(con, ...) %>% ...` chain)
`...`	Other arugments passed to `bigrquery::bq_table_download()` when `collect = TRUE`
`con`	Connection to the allofus SQL database. Defaults to `getOption("aou.default.con")`, which is created automatically with `aou_connect()`.

Details

[Experimental]

Value

A reference to a temporary table in the database.

Examples

```
con <- aou_connect()
tmp_tbl <- dplyr::tbl(con, "concept") %>%
  dplyr::select(concept_id) %>%
  head(10) %>%
  aou_compute()

tmp_tbl
```

aou_concept_set *Get occurrences of a concept set from AoU for a given cohort*

Description

Retrieves occurrences of a concept set from the All of Us database for a given cohort.

Usage

```
aou_concept_set(
  cohort = NULL,
  concepts,
  start_date = NULL,
  end_date = NULL,
 domains = c("condition", "measurement", "observation", "procedure", "drug", "device",
    "visit"),
  output = "indicator",
  concept_set_name = "concept_set",
  min_n = 1,
  collect = FALSE,
  ...,
  con = getOption("aou.default.con")
)
```

Arguments

`cohort`	Reference to a remote table or local dataframe with a column called "person_id", and (possibly) columns for `start_date` and `end_date`. If not provided, defaults to entire All of Us cohort.
`concepts`	a vector of concept ids
`start_date`	chr; the name of the start_date column in the cohort table; defaults to NULL to pull data across all dates
`end_date`	chr; the name of the end_date column in the cohort table; defaults to NULL to pull data across all dates
`domains`	chr; a vector of domains to search for the concepts in ("condition", "measurement", "observation", "procedure", "drug", "device", "visit"); defaults to all
`output`	one of "indicator", "count", "all"; do you want to return a 1 if a person has any matching concepts and 0 if not ("indicator"), the number of matching concepts per person ("count"), or all info about the matching concepts ("all"). Defaults to "indicator"
`concept_set_name`	chr; If output = "indicator" or output = "n", name for that column. Defaults to "concept_set".
`min_n`	dbl; If output = "indicator", the minimum number of occurrences per person to consider the indicator true. Defaults to 1.

`collect`	Whether to bring the resulting table into local memory (`collect = TRUE`) as a dataframe or leave as a reference to a database table (for continued analysis using, e.g., `dbplyr`). Defaults to `FALSE`.
`...`	further arguments passed along to `collect()` if `collect = TRUE`
`con`	Connection to the allofus SQL database. Defaults to `getOption("aou.default.con")`, which is created automatically with `aou_connect()`.

Value

A dataframe if `collect = TRUE`; a reference to a remote database table if not.

Examples

```
# indicator for any aspirin at any time
aspirin_users <- aou_concept_set(dplyr::tbl(con, "person"),
  concepts = 1191, concept_set_name = "aspirin", domains = "drug"
)

# starting with person table to create a cohort
people <- dplyr::tbl(con, "person") %>%
  dplyr::filter(person_id < 2000000) %>%
  dplyr::mutate(
    start = as.Date("2021-01-01"),
    end = as.Date("2023-12-31")
  )

dat <- aou_concept_set(
  cohort = people,
  concepts = c(725115, 1612146, 1613031),
  start_date = "start",
  end_date = "end",
  concept_set_name = "CGM",
  output = "all"
)
```

`aou_connect` *Create a connection to the database in All of Us*

Description

Connects to the All of Us database and returns a BigQueryConnection object. You can reference this object to query the database using R and or SQL code. A message is printed with the connection status (successful or not).

Usage

```
aou_connect(CDR = getOption("aou.default.cdr"), ...)
```

Arguments

`CDR`	The name of the "curated data repository" to connect to. Defaults to `getOption("aou.default.cdr")`, which is `Sys.getenv('WORKSPACE_CDR')` if not specified otherwise (i.e., the "mainline" CDR). On the controlled tier, specify the "base" CDR with `CDR = paste0(Sys.getenv('WORKSPACE_CDR'), "_base")`.
`...`	Further arguments passed along to `DBI::dbConnect()`.

Details

You can reference this object to connect to the All of Us database and run SQL code using, e.g., `dbplyr` or `DBI`. A message is printed with the connection status (successful or not).

Value

A `BigQueryConnection` object. This object is also saved as an option (`getOption("aou.default.con")`).

Examples

```
con <- aou_connect()
# reference the observation table in the database
dplyr::tbl(con, "observation")
# print a list of the tables in the database
DBI::dbListTables(con)
```

`aou_create_temp_table` *Creates a temporary table from a local data frame or tibble*

Description

Experimental function that builds a local tibble into an SQL query and generates a temporary table. Larger tables will be broken up into consequitive SQL queries; making `nchar_batch` smaller can avoid errors but will take longer. The table will only exist for the current connection session and will need to be created again in a new session.

Usage

```
aou_create_temp_table(
  data,
  nchar_batch = 1e+06,
  ...,
  con = getOption("aou.default.con")
)
```

Arguments

`data`	A local dataframe (or tibble)
`nchar_batch`	approximate number of characters to break up each SQL query
`...`	Not currently used
`con`	Connection to the allofus SQL database. Defaults to `getOption("aou.default.con")`, which is created automatically with `aou_connect()`.

Details

[Experimental]

Value

a reference to a temporary table in the database with the data from `df`

Examples

```
con <- aou_connect()
df <- data.frame(
  concept_id = c(
    439331, 4290245, 42535816, 46269813,
    2784565, 45765502, 434112, 4128031, 435640, 45876808
  ),
  category = c(
    "AB", "DELIV", "DELIV", "SA", "DELIV",
    "LB", "DELIV", "DELIV", "PREG", "SA"
  ),
  gest_value = c(NA, NA, NA, NA, NA, NA, NA, NA, 25, NA)
)
tmp_tbl <- aou_create_temp_table(df)
```

`aou_health_history` *All of Us Health History Codebook*

Description

This table consists of rows of the codebook pertaining to the health history questions. In early All of Us surveys, these questions were asked separately about the respondent and the respondent's family. In the current version, the questions are asked on the same survey. The nested nature of these questions makes them challenging to deal with. It can also be accessed in R using `allofus::aou_health_history`.

- Code to generate table

Usage

```
aou_health_history
```

Format

`aou_health_history` A data frame with 1685 rows and 9 columns:

question chr; Question asked on survey

relative chr; Person to whom the answer pertains

condition chr; Formatted text name of concept

category chr; Type of health condition

concept_code chr; Concept code from AOU codebook

concept_id_specific int; Concept id for the answer

concept_id_overall int; Concept id for the condition overall

concept_id_question int; Concept id for the overarching question

form_name chr; Survey name

`aou_join` *Join current query to another table*

Description

Joins two tables in the All of Us database. A less verbose wrapper for the dplyr::*_join() functions with some added safeguards.

Usage

```
aou_join(
  data,
  table,
  type,
  by = NULL,
  suffix = c("_x", "_y"),
  x_as = NULL,
  y_as = NULL,
  ...,
  con = getOption("aou.default.con")
)
```

Arguments

`data`	unexecuted SQL query from dbplyr/dplyr.
`table`	the omop table (or other remote table in your schema) you wish to join, as a character string, or a tbl object.
`type`	the type of join; types available in dplyr: "left", "right", "inner", "anti", "full", etc.
`by`	columns to join on

`suffix`	suffix preferences to add when joining data with the same column names not specified in the by argument.
`x_as`	optional; a string for the name of the left table
`y_as`	optional; a string for the name of the right table
`...`	Additional arguments passed on to the join function
`con`	Connection to the allofus SQL database. Defaults to `getOption("aou.default.con")`, which is created automatically with `aou_connect()`.

Details

There are a few good reasons to use aou_join() when possible over the x_join functions from dplyr. First, it reduces the code necessary to join an existing table to another table. Second, it includes checks/workarounds for two sources of common errors using dbplyr: it automatically appends the x_as and y_as arguments to the join call if they are not provided and it changes the default suffix from .x/.y to _x/_y for cases with shared column names not specified by the by argument which will result in a SQL error.

Value

Reference to the remote table created by the join.

Examples

```
con <- aou_connect()
obs_tbl <- dplyr::tbl(con, "observation") %>%
  dplyr::select(-provider_id)
obs_tbl %>%
  aou_join("person", type = "left", by = "person_id")
```

`aou_ls_bucket` *List the current files in your bucket*

Description

Lists all files in the bucket or files matching a certain pattern.

Usage

```
aou_ls_bucket(
  pattern = "",
  silent = FALSE,
  recursive = TRUE,
  bucket = getOption("aou.default.bucket"),
  gsutil_args = ""
)
```

Arguments

`pattern`	Regular expression, such as "*.csv" or a single file name e.g., "mydata.csv". Default will find all files apart from notebooks (.ipynb files).
`silent`	Whether to omit the names of files found. Defaults to `FALSE`.
`recursive`	Whether to search subdirectories. Defaults to `TRUE`.
`bucket`	Bucket to retrieve file from. Defaults to `getOption("aou.default.bucket")`, which is `Sys.getenv('WORKSPACE_BUCKET')` unless specified otherwise.
`gsutil_args`	A string containing other arguments passed to `gsutil ls`. See `https://cloud.google.com/storage/docs/gsutil/commands/ls` for details.

Value

A vector of file names

Examples

```
# list all files, including in subdirectories
aou_ls_bucket()
# list all csv files
aou_ls_bucket("*.csv")
```

`aou_ls_workspace` *List the current files in your workspace*

Description

Lists all data files in the workspace or files matching a certain pattern.

Usage

```
aou_ls_workspace(pattern = "", silent = FALSE, ...)
```

Arguments

`pattern`	Regular expression, such as "*.csv" or a single file name e.g., "mydata.csv". Default will find all files apart from notebooks (.ipynb, .Rmd, .qmd files).
`silent`	Whether to omit the names of files found. Defaults to `FALSE`.
`...`	Other arguments passed to `list.files()`

Value

A vector of file names

Examples

```
my_workspace_files <- aou_ls_workspace(silent = TRUE)
aou_ls_workspace("*.csv")
aou_ls_workspace(path = "data")
```

---

aou_observation_period

Generate an observation period table

Description

Generates a temporary observation period table based the first and last event in the electronic medical record data. Because some EHR sites have contributed data from several decades ago, researchers might want to consider further constraining this table to reasonable date ranges of interest (e.g., setting all observation_period_start_date values to no earlier than 01/01/2010).

Usage

```
aou_observation_period(
  cohort = NULL,
  collect = FALSE,
  ...,
  con = getOption("aou.default.con")
)
```

Arguments

`cohort`	Reference to a remote table or local dataframe with a column called "person_id"
`collect`	Whether to bring the resulting table into local memory (`collect = TRUE`) as a dataframe or leave as a reference to a database table (for continued analysis using, e.g., `dbplyr`). Defaults to `FALSE`.
`...`	Further arguments passed along to `collect()` if `collect = TRUE`
`con`	Connection to the allofus SQL database. Defaults to getOption("aou.default.con"), which is set automatically if you use `aou_connect()`

Details

[Experimental]

The current observation period table in the All of Us OMOP CDM is not always appropriate for cohorts generated using OHDSI tools such as ATLAS. Some observation periods are overly short and some participants have hundreds of observation periods.

This function generates an observation period table from the first occurrence of a clinical event in the EHR tables to the last clinical event in the EHR tables. It will only return a single observation period per person_id in the database. If `collect = FALSE`, the function returns a query to a temporary table in the database which can be referenced by typical dplyr functions.

Normal OMOP conventions for EHR suggest that long lapses of time between clinical events may indicate that the person was not "observed" during this period. However, due to the diverse nature of clinical EHR data contributed to All of Us, it seems most conservative to assume that the person was observed from their first to last clinical event. See https://ohdsi.github.io/CommonDataModel/ehrObsPeriods.html for more details.

Some users have clinical events going back to before the time of widespread electronic medical record use (e.g., the 1980s and 1990s). This function considers all EHR data in the database, regardless of the date of the clinical event, but we recommend that users consider the implications of including data from the 1980s and 1990s. It may be more prudent to exclude data prior to a more recent cutoff date so that the EHR data is more likely to be accurate, though this decision depends highly on the research question (see example below).

Users should note that the aou_observation_period function will only generate observation periods for participants who have at least one clinical observation. If participant in the AllofUs research program who did not include electronic health record data are included in the cohort argument, or elected to contribute data but have no data to contribute, they will not be included in the generated observation period table.

Value

A dataframe if `collect = TRUE`; a reference to a remote database table if not. Columns will be "person_id", "observation_period_start_date", and "observation_period_end_date".

Examples

```
library(dplyr)
con <- aou_connect()

# create observation_period table for everyone
observation_period_tbl <- aou_observation_period()

# create a cohort of participants with EHR data and at least one year
# of observation before they took the first survey

# first, create an index date as the first date a survey was taken
index_date_tbl <- tbl(con, "ds_survey") %>%
  group_by(person_id) %>%
  summarize(index_date = as.Date(min(survey_datetime, na.rm = TRUE)),
            .groups = "drop")

# join with observation_period_tbl
cohort <- tbl(con, "cb_search_person") %>%
  filter(has_ehr_data == 1) %>%
  inner_join(index_date_tbl, by = "person_id") %>%
  inner_join(observation_period_tbl, by = "person_id") %>%
  filter(
    observation_period_start_date <= DATE_ADD(
      index_date,
      sql(paste0("INTERVAL ", -1, " year"))
    ),
    index_date <= observation_period_end_date
  ) %>%
  select(person_id, gender, sex_at_birth,
    race, ethnicity, age_at_consent,
    index_date, observation_period_start_date, observation_period_end_date)

# head(cohort)

# create an observation period table with a minimum start date (e.g., 2010-01-01)
# to only look at EHR data after that date
observation_period_tbl %>%
  mutate(
    observation_period_start_date =
      if_else(observation_period_start_date < as.Date("2010-01-01"),
        as.Date("2010-01-01"),
        observation_period_start_date
      )
  ) %>%
  filter(observation_period_end_date > as.Date("2010-01-01"))
```

aou_session_info *Print session information for the AoU R environment*

Description

Returns a table of information that is necessary to fully reproduce your analyses. Specifically, it includes R version, the packages loaded and their versions, and the All of Us CDR release that you are using.

Usage

```
aou_session_info(CDR = getOption("aou.default.cdr"))
```

Arguments

CDR	The name of the CDR to use. Defaults to `getOption("aou.default.cdr")`

Value

A list with three elements: the platform, the AoU release, and the packages

Examples

```
allofus::aou_session_info()
```

aou_sql *Execute a SQL query on the All of Us database*

Description

Executes an SQL query on the All of Us database

Usage

```
aou_sql(
  query,
  collect = FALSE,
  debug = FALSE,
  ...,
  con = getOption("aou.default.con"),
  CDR = getOption("aou.default.cdr")
)
```

Arguments

`query`	A SQL query (BigQuery dialect) to be executed. Interpreted with `glue::glue()`, so expressions enclosed with braces will be evaluated. References to `"{CDR}"` or `"{cdr}"` will be evaluated automatically (see examples).
`collect`	Whether to bring the resulting table into local memory (`collect = TRUE`) as a dataframe or leave as a reference to a database table (for continued analysis using, e.g., `dbplyr`). Defaults to `FALSE`.
`debug`	Print the query to the console; useful for debugging.
`...`	All other arguments passed to `bigrquery::bq_table_download()` if `collect = TRUE`.
`con`	Connection to the allofus SQL database. Defaults to `getOption("aou.default.con")`, which is created automatically with `aou_connect()`. Only needed if `collect = FALSE`.
`CDR`	The name of the "curated data repository" that will be used in any references of the form `"{CDR}"` or `"{cdr}"` in the query (see examples). Defaults to `getOption("aou.default.cdr")`, which is `Sys.getenv('WORKSPACE_CDR')` if not specified otherwise (i.e., the "mainline" CDR). On the controlled tier, specify the "base" CDR with `CDR = paste0(Sys.getenv('WORKSPACE_CDR'), "_base")`.

Value

A dataframe if `collect = TRUE`; a reference to a remote database table if not.

Examples

```
# Examples based on AoU snippets
aou_sql("
  -- Compute the count of unique participants in our All of Us cohort.
  SELECT
  COUNT(DISTINCT person_id) AS total_number_of_participants
  FROM
  `{CDR}.person`
", collect = TRUE)

MEASUREMENT_OF_INTEREST <- "hemoglobin"
aou_sql('
-- Compute summary information for our measurements of interest for our cohort.
--
-- PARAMETERS:
--  MEASUREMENT_OF_INTEREST: a case-insensitive string, such as "hemoglobin", to be compared
--                          to all measurement concept names to identify those of interest

WITH
  --
  -- Use a case insensitive string to search the measurement concept names of those
  -- measurements we do have in the measurements table.
  --
  labs_of_interest AS (
  SELECT
    measurement_concept_id,
    measurement_concept.concept_name AS measurement_name,
    unit_concept_id,
    unit_concept.concept_name AS unit_name
  FROM
    `{CDR}.measurement`
  LEFT JOIN `{CDR}.concept` AS measurement_concept
  ON measurement_concept.concept_id = measurement_concept_id
  LEFT JOIN `{CDR}.concept` AS unit_concept
  ON unit_concept.concept_id = unit_concept_id
  WHERE
    REGEXP_CONTAINS(measurement_concept.concept_name, r"(?i){MEASUREMENT_OF_INTEREST}")
  GROUP BY
    measurement_concept_id,
    unit_concept_id,
    measurement_concept.concept_name,
    unit_concept.concept_name
)
  --
  -- Summarize the information about each measurement concept of interest that our
  -- prior query identified.
  --
SELECT
  measurement_name AS measurement,
  IFNULL(unit_name, "NA") AS unit,
  COUNT(1) AS N,
  COUNTIF(value_as_number IS NULL
    AND (value_as_concept_id IS NULL
      OR value_as_concept_id = 0)) AS missing,
  MIN(value_as_number) AS min,
  MAX(value_as_number) AS max,
  AVG(value_as_number) AS avg,
  STDDEV(value_as_number) AS stddev,
  APPROX_QUANTILES(value_as_number, 4) AS quantiles,
  COUNTIF(value_as_number IS NOT NULL) AS num_numeric_values,
  COUNTIF(value_as_concept_id IS NOT NULL
      AND value_as_concept_id != 0) AS num_concept_values,
  COUNTIF(operator_concept_id IS NOT NULL) AS num_operators,
  IF(src_id = "PPI/PM", "PPI", "EHR") AS measurement_source,
  measurement_concept_id,
  unit_concept_id
FROM
  `{CDR}.measurement`
INNER JOIN
 labs_of_interest USING(measurement_concept_id, unit_concept_id)
LEFT JOIN
  `{CDR}.measurement_ext` USING(measurement_id)
GROUP BY
  measurement_concept_id,
  measurement_name,
  measurement_source,
  unit_concept_id,
  unit_name
ORDER BY
  N DESC
', collect = TRUE)
```

aou_survey — *Function to query allofus observation table for survey responses*

Description

Extracts survey responses in a tidy format that also includes ‘skip’ responses and collapses across all versions of the person health / personal medical history surveys. Currently responses in the ‘ds_survey’ table omit skipped responses. Responses are returned as Yes" if the respondent answered that the individual had the condition, No" if the respondent answered that the individual did not have that condition (or omitted it when selecting from related conditions), a skip response if the question was skipped, and NA if the respondent did not answer the question. Returns a data frame or SQL tbl with the initial cohort table along with a column for each question included in questions and answers foreach person_id in the cells. To find the desired survey questions, use the all of us data dictionary, survey codebook, Athena, data browser, or the modified codebook which can be found in the allofus R package.

Usage

```
aou_survey(
  cohort = NULL,
  questions,
  question_output = "concept_code",
  clean_answers = TRUE,
  collect = FALSE,
  ...,
  con = getOption("aou.default.con")
)
```

Arguments

`cohort`	Reference to a remote table or local dataframe with a column called "person_id"
`questions`	either a vector of concept_ids or concept_codes for questions to return results
`question_output`	how to name the columns. Options include as the text of the concept code ("concept_code"), as concept ids preceded by "x_" ("concept_id"), or using a custom vector of column names matching the vector of `questions`. Defaults to "concept_code".
`clean_answers`	whether to clean the answers to the survey questions. Defaults to TRUE.
`collect`	Whether to bring the resulting table into local memory (`collect = TRUE`) as a dataframe or leave as a reference to a database table (for continued analysis using, e.g., `dbplyr`). Defaults to `FALSE`.
`...`	additional arguments passed to `collect()` when `collect = TRUE`
`con`	connection to the allofus SQL database. Defaults to getOption("aou.default.con"), which is created automatically with `aou_connect()`

Details

The function will return a dataframe or SQL tbl with the initial cohort table along with a column for each question included in `questions` and answers for each person_id in the cells. The column names (questions) can be returned as the concept_code or concept_id or by providing new column names. For each question, a column with the suffix "_date" is included with the date on which the question was answered. When questions can have multiple answers ("checkbox"-style questions), answers are returned as a comma-separated string.

To find the desired survey questions, use the all of us data dictionary, survey codebook, athena, data browser, or the allofus R package modified codebook which can be found here: https://roux-ohdsi.github.io/allofus/vignettes/searchable_codebook.html For questions regarding an individual's health history or family health history, the function requires the specific concept_id (or concept_code) for individual in question, whether that is "self" or another relative. Responses are returned as "Yes" if the respondent answered that the individual had the condition, "No" if the respondent answered that the individual did not have that condition (or omitted it when selecting from related conditions), a skip response if the question was skipped, and NA if the respondent did not answer the question.

Value

A dataframe if `collect` = `TRUE`; a reference to a remote database table if not.

Examples

```
con <- aou_connect()
cohort <- dplyr::tbl(con, "person") %>%
  dplyr::filter(person_id > 5000000) %>%
  dplyr::select(person_id, year_of_birth, gender_concept_id)

aou_survey(
  cohort,
  questions = c(1585375, 1586135),
  question_output = "concept_code"
)
aou_survey(
  cohort,
  questions = c(1585811, 1585386),
  question_output = c("pregnancy", "insurance")
)
aou_survey(
  cohort,
  questions = c(1585375, 1586135, 1740719, 43529932),
  question_output = c("income", "birthplace", "grandpa_bowel_obstruction", "t2dm"),
  collect = FALSE
)

aou_survey(cohort,
  questions = 1384452,
  question_output = "osteoarthritis"
) %>%
  dplyr::count(osteoarthritis)
```

`aou_tables` *List tables in the AoU Database*

Description

Prints a list of all of the tables in the All of Us Big Query Database.

Usage

```
aou_tables(remove_na = TRUE, ..., con = getOption("aou.default.con"))
```

Arguments

`remove_na`	Whether to remove tables that are not in the data dictionary. Defaults to `TRUE`
`...`	Not currently used
`con`	Connection to the allofus SQL database. Defaults to `getOption("aou.default.con")`, which is created automatically with `aou_connect()`.

Value

A dataframe with the table names and the number of columns

Examples

```
con <- aou_connect()
aou_tables()
```

`aou_table_info` *Table of tables, columns, and use for researchers from the CT data dictionary*

Description

A data from with rows of the All of Us codebook pertaining to the health history questions. In early All of Us surveys, these questions were asked separately about the respondent and the respondent's family. In the current version, the questions are asked on the same survey. The nested nature of these questions can make them challenging to extract and analyze.

- Code to generate table

Usage

```
aou_table_info
```

Format

`aou_table_info`

table_name chr; name of the table

columns chr; columns in the table

recommended_for_research chr; whether the table is recomended for research

aou_workspace_to_bucket

Save a file from your workspace to your bucket

Description

Moves a file saved in on the persistent disk to the workspace bucket, where it can be stored even if a compute environment is deleted.

Usage

```
aou_workspace_to_bucket(
  file,
  directory = FALSE,
  bucket = getOption("aou.default.bucket")
)
```

Arguments

`file`	The name of a file in your bucket, a vector of multiple files, a directory, or a file pattern (e.g. ".csv"). See Details.
`directory`	Whether `file` refers to an entire directory you want to move.
`bucket`	Bucket to save files to. Defaults to `getOption("aou.default.bucket")`, which is `Sys.getenv('WORKSPACE_BUCKET')` unless specified otherwise.

Details

This function moves a file saved in a workspace to a bucket, where it can be retrieved even if the environment is deleted. To use, first save the desired object as a file to the workspace (e.g., `write.csv(object, "filename.csv")`) and then run this function (e.g., `aou_workspace_to_bucket(files = "filename.csv")`). See `https://cloud.google.com/storage/docs/gsutil/commands/cp` for details on the underlying function.

Value

Nothing

Examples

```
# create test files in a temporary directory
tmp <- tempdir()
write.csv(data.frame(x = 1), file.path(tmp, "testdata1.csv"))
write.csv(data.frame(y = 2), file.path(tmp, "testdata2.csv"))
# save a file to the bucket
aou_workspace_to_bucket(file.path(tmp, "testdata1.csv"))
# save multiple files at once
aou_workspace_to_bucket(c(file.path(tmp, "testdata1.csv"), file.path(tmp, "testdata2.csv")))
# save an entire directory
```

```
aou_workspace_to_bucket(tmp, directory = TRUE)
```

Index