Package 'rcprd'

November 13, 2024

Title Extraction and Management of Clinical Practice Research Datalink Data

Version 0.0.1

Description Simplify the process of extracting and processing Clinical Practice Research Datalink (CPRD) data in order to build datasets ready for statistical analysis. This process is difficult in 'R', as the raw data is very large and cannot be read into the R workspace. 'rcprd' utilises 'RSQLite' to create 'SQLite' databases which are stored on the hard disk. These are then queried to extract the required information for a cohort of interest, and create datasets ready for statistical analysis. The processes follow closely that from the 'rEHR' package, see Springate et al., (2017) <doi:10.1371/journal.pone.0171784>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Depends data.table

Imports dplyr, fastmatch, RSQLite, stringr

Config/testthat/edition 3

URL https://alexpate30.github.io/rcprd/

NeedsCompilation no

Author Alexander Pate [aut, cre, cph] (<https://orcid.org/0000-0002-0849-3458>)

Maintainer Alexander Pate <alexander.pate@manchester.ac.uk>

Repository CRAN

Date/Publication 2024-11-13 14:00:02 UTC

Contents

`add_to_database` *Adds a single .txt file to an SQLite database on the hard disk.*

Description

Add the raw data from one of the CPRD flatfiles to an SQLite database.

Usage

```
add_to_database(
  filepath,
  filetype = c("observation", "drugissue", "referral", "problem", "consultation",
    "hes_primary", "death"),
  nrows = -1,
  select = NULL,
  subset_patids = NULL,
  use_set = FALSE,
  db,
  extract_txt_func = NULL,
  tablename = NULL,
  ...
)
```

Arguments

filepath	Path to .txt file on your system.
filetype	Type of CPRD Aurum file (observation, drugissue, referral, problem, consultation, hes_primary, death)
nrows	Number of rows to read in from .txt file.
select	Character vector of column names to select before adding to the SQLite database.
subset_patids	Patient id's to subset the .txt file on before adding to the SQLite database.
use_set	Reduce subset_patids to just those with a corresponding set value to the .txt file being read in. Can greatly improve computational efficiency when subset_patids is large. See vignette XXXX for more details.
db	An open SQLite database connection created using RSQLite::dbConnect.
extract_txt_func	User-defined function to read the .txt file into R.
tablename	Name of table in SQLite database that the data will be added to.
...	Extract arguments passed to read.table (or extract_txt_func) when reading in .txt files.

Details

Will add the file to a table named `filetype` in the SQLite database, unless `tablename` is specified.

If `use_set = FALSE`, then `subset_patids` should be a vector of patid's that the .txt files will be subsetted on before adding to the SQLite database. If `use_set = TRUE`, then `subset_patids` should be a dataframe with two columns, `patid` and `set`, where `set` corresponds to the number in the file name following the word 'set'. This functionality is provided to increase computational efficiency when subsetting to a cohort of patients which is very large (millions). This can be a computationally expensive process as each flatfile being read in, must be cross matched with a large vector . The CPRD flatfiles are split up into groups which can be identified from their naming convention. Patients from set 1, will have their data in DrugIssue, Observation, etc, all with the same "set" suffix in the flatfile name. We can utilise this to speed up the process of subsetting the data from the flatfiles

to only those with patids in subset.patid. Instead we subset to those with patids in subset_patids, and with the corresponding value of "set", which matches the suffix "set" in the CPRD flatfile filename. For example, patients in the Patient file which had suffix "set1", will have their medical data in the Observation file with suffix "set1". When subsetting the Observation file to those in subset_patids (our cohort), we only need to do so for patients who were also in the patient file with suffix "set1". If the cohort of patients for which you want to subset the data to is very small, the computational gains from this argument are minor and it can be ignored.

The function for reading in the .txt file will be chosen from a set of functions provided with rcprd, based on the fletype (`filetype`). `extract_txt_func` does not need to be specified unless wanting to manually define the function for doing this. This may be beneficial if wanting to change variable formats, or if the variables in the .txt files change in future releases of CPRD AURUM.

Value

Adds .txt file to SQLite database on hard disk.

Examples

```
## Create connection to a temporary database
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Add observation data
add_to_database(filepath = system.file("aurum_data",
"aurum_allpatid_set1_extract_observation_001.txt", package = "rcprd"),
filetype = "observation", db = aurum_extract, overwrite = TRUE)

## Query database
RSQLite::dbGetQuery(aurum_extract, 'SELECT * FROM observation', n = 3)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

---

combine_query *Combine a database query with a cohort.*

Description

An S3 generic function that can be used on database queries from Aurum or GOLD extracts. Combine a database query with a cohort, only retaining observations between `time_prev` days prior to `indexdt`, and `time_post` days after `indexdt`, and for test data with values between `lower_bound` and `upper_bound`. The most recent `numobs` observations will be returned. `cohort` must contain variables `patid` and `indexdt`. The `type` of query must be specified for appropriate data manipulation. Input `type = med` if interested in medical diagnoses from the observation file, and `type = test` if interseted in test data from the observation file.

Usage

```
combine_query(
  db_query,
  cohort,
  query_type = c("med", "drug", "test", "hes_primary", "death"),
  time_prev = Inf,
  time_post = Inf,
  lower_bound = -Inf,
  upper_bound = Inf,
  numobs = 1,
  value_na_rm = TRUE,
  earliest_values = FALSE,
  reduce_output = TRUE
)
```

Arguments

db_query	Output from database query (ideally obtained through db_query).
cohort	Cohort to combine with the database query.
query_type	Type of query
time_prev	Number of days prior to index date to look for codes.
time_post	Number of days after index date to look for codes.
lower_bound	Lower bound for returned values when query_type = "test".
upper_bound	Upper bound for returned values when query_type = "test".
numobs	Number of observations to be returned.
value_na_rm	If TRUE will remove data with NA in the value column of the queried data and remove values outside of lower_bound and upper_bound when query_type = "test".
earliest_values	If TRUE will return the earliest values as opposed to most recent.
reduce_output	If TRUE will reduce output to just patid, event date, medical/product code, and test value.

Details

value_na_rm = FALSE may be of use when extracting variables like smoking status, where we want test data for number of cigarettes per day, but do not want to remove all observations with NA in the value column, because the medcodeid itself may indicate smoking status.

Value

A data.table with observations that meet specified criteria.

Examples

```
## Create connection to a temporary database
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Add observation data from all observation files in specified directory
cprd_extract(db = aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation")

## Query database for a specific medcode
db_query <- db_query(db_open = aurum_extract,
tab ="observation",
codelist_vector = "187341000000114")

## Define cohort
pat<-extract_cohort(filepath = system.file("aurum_data", package = "rcprd"))

### Add an index date to pat
pat$indexdt <- as.Date("01/01/2020", format = "%d/%m/%Y")

## Combine query with cohort retaining most recent three records
combine_query(cohort = pat,
db_query = db_query,
query_type = "med",
numobs = 3)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

`combine_query.aurum` *Combine a CPRD aurum database query with a cohort.*

Description

An S3 method that can be used on database queries from Aurum extracts. Combine a database query with a cohort, only retaining observations between `time_prev` days prior to `indexdt`, and `time_post` days after `indexdt`, and for test data with values between `lower_bound` and `upper_bound`. The most recent `numobs` observations will be returned. `cohort` must contain variables `patid` and `indexdt`. The `type` of query must be specified for appropriate data manipulation. Input `type = med` if interested in medical diagnoses from the observation file, and `type = test` if interseted in test data from the observation file.

Usage

```
## S3 method for class 'aurum'
combine_query(
  db_query,
```

```
  cohort,
  query_type,
  time_prev = Inf,
  time_post = Inf,
  lower_bound = -Inf,
  upper_bound = Inf,
  numobs = 1,
  value_na_rm = TRUE,
  earliest_values = FALSE,
  reduce_output = TRUE
)
```

Arguments

db_query	Output from database query (ideally obtained through db_query).
cohort	Cohort to combine with the database query.
query_type	Type of query
time_prev	Number of days prior to index date to look for codes.
time_post	Number of days after index date to look for codes.
lower_bound	Lower bound for returned values when query_type = "test".
upper_bound	Upper bound for returned values when query_type = "test".
numobs	Number of observations to be returned.
value_na_rm	If TRUE will remove data with NA in the value column of the queried data and remove values outside of lower_bound and upper_bound when query_type = "test".
earliest_values	If TRUE will return the earliest values as opposed to most recent.
reduce_output	If TRUE will reduce output to just patid, event date, medical/product code, and test value.

Details

value_na_rm = FALSE may be of use when extracting variables like smoking status, where we want test data for number of cigarettes per day, but do not want to remove all observations with NA in the value column, because the medcodeid itself may indicate smoking status.

Value

A data.table with observations that meet specified criteria.

Examples

```
## Create connection to a temporary database
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Add observation data from all observation files in specified directory
cprd_extract(db = aurum_extract,
```

```
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation")

## Query database for a specific medcode
db_query <- db_query(db_open = aurum_extract,
tab ="observation",
codelist_vector = "187341000000114")

## Define cohort
pat<-extract_cohort(filepath = system.file("aurum_data", package = "rcprd"))

### Add an index date to pat
pat$indexdt <- as.Date("01/01/2020", format = "%d/%m/%Y")

## Combine query with cohort retaining most recent three records
combine_query(cohort = pat,
db_query = db_query,
query_type = "med",
numobs = 3)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

`combine_query_boolean` *Combine a database query with a cohort returning a 0/1 vector depending on whether each individual has a recorded code of interest.*

Description

An S3 generic function that can be used on database queries from Aurum or GOLD extracts. Combine a database query with a cohort returning a 0/1 vector depending on whether each individual has a recorded code of interest. `cohort` must contain variables `patid` and `indexdt`. The database query will be merged with the cohort by variable `patid`. If an individual has at least `numobs` observations between `time_prev` days prior to `indexdt`, and `time_post` days after `indexdt`, a 1 will be returned, 0 otherwise. The `type` of query must be specified for appropriate data manipulation.

Usage

```
combine_query_boolean(
  db_query,
  cohort,
  query_type = c("med", "drug"),
  time_prev = Inf,
  time_post = 0,
  numobs = 1
)
```

Arguments

db_query	Output from database query (ideally obtained through db_query).
cohort	Cohort to combine with the database query.
query_type	Type of query
time_prev	Number of days prior to index date to look for codes.
time_post	Number of days after index date to look for codes.
numobs	Number of observations required to be observed in specified time window to return a 1.

Value

A 0/1 vector.

---

combine_query_boolean.aurum

Combine a CPRD aurum database query with a cohort returning a 0/1 vector depending on whether each individual has a recorded code of interest.

Description

An S3 method that can be used on database queries from Aurum extracts. Combine a database query with a cohort returning a 0/1 vector depending on whether each individual has a recorded code of interest. cohort must contain variables patid and indexdt. The database query will be merged with the cohort by variable patid. If an individual has at least numobs observations between time_prev days prior to indexdt, and time_post days after indexdt, a 1 will be returned, 0 otherwise. The type of query must be specified for appropriate data manipulation.

Usage

```
## S3 method for class 'aurum'
combine_query_boolean(
  db_query,
  cohort,
  query_type,
  time_prev = Inf,
  time_post = 0,
  numobs = 1
)
```

Arguments

db_query	Output from database query (ideally obtained through db_query).
cohort	Cohort to combine with the database query.
query_type	Type of query

time_prev	Number of days prior to index date to look for codes.
time_post	Number of days after index date to look for codes.
numobs	Number of observations required to be observed in specified time window to return a 1.

Value

A 0/1 vector.

Examples

```
## Create connection to a temporary database
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Add observation data from all observation files in specified directory
cprd_extract(db = aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation")

## Query database for a specific medcode
db_query <- db_query(db_open = aurum_extract,
tab ="observation",
codelist_vector = "187341000000114")

## Define cohort
pat<-extract_cohort(filepath = system.file("aurum_data", package = "rcprd"))

### Add an index date to pat
pat$indexdt <- as.Date("01/01/2020", format = "%d/%m/%Y")

## Combine query with cohort retaining most recent three records
combine_query_boolean(cohort = pat,
db_query = db_query,
query_type = "med",
numobs = 3)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

---

connect_database *Open connection to SQLite database*

Description

Open connection to SQLite database

Usage

```
connect_database(dbname)
```

Arguments

dbname	Name of SQLite database on hard disk (including full file path relative to working directory)

Value

No return value, called to open a database connection.

Examples

```
## Connect to a database
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Check connection is open
inherits(aurum_extract, "DBIConnection")

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

---

cprd_extract *Adds all the .txt files in a directory, with certain file names, to an SQLite database on the hard disk.*

Description

Add the raw data from more than one of the CPRD flatfiles to an SQLite database.

Usage

```
cprd_extract(
  db,
  filepath,
  filetype = c("observation", "drugissue", "referral", "problem", "consultation",
    "hes_primary", "death"),
  nrows = -1,
  select = NULL,
  subset_patids = NULL,
  use_set = FALSE,
  extract_txt_func = NULL,
  str_match = NULL,
  tablename = NULL
)
```

Arguments

db	An open SQLite database connection created using RSQLite::dbConnect.
filepath	Path to directory containing .txt files.
filetype	Type of CPRD Aurum file (observation, drugissue, referral, problem, consultation, hes_primary, death)
nrows	Number of rows to read in from .txt file.
select	Vector of column names to select before adding to the SQLite database.
subset_patids	Patient id's to subset the .txt file on before adding to the SQLite database.
use_set	Reduce subset_patids to just those with a corresponding set value to the .txt file being read in. Can greatly improve computational efficiency when subset_patids is large. See vignette XXXX for more details.
extract_txt_func	User-defined function to read the .txt file into R.
str_match	Character vector to match on when searching for file names to add to the database.
tablename	Name of table in SQLite database that the data will be added to.

Details

By default, will add files that contain `filetype` in the file name to a table named `filetype` in the SQLite database. If `str_match` is specified, will add files that contain `str_match` in the file name to a table named `str_match` in the SQLite database. In this case, `filetype` will still be used to choose which function reads in and formats the raw data, although this can be overwritten with `extract_txt_func`. If argument `tablename` is specified, data will be added to a table called `tablename` in the SQLite database.

Currently, rcprd only deals with `filetype = c("observation", "drugissue", "referral", "problem", "consultation", "hes_primary", "death")` by default. However, by using `str_match` and `extract_txt_func`, the user can manually search for files with any string in the file name, and read them in and format using a user-defined function. This means the user is not restricted to only adding the pre-defined file types to the SQLite database.

If `use_set = FALSE`, then `subset_patids` should be a vector of patid's that the .txt files will be subsetted on before adding to the SQLite database. If `use_set = TRUE`, then `subset_patids` should be a dataframe with two columns, `patid` and `set`, where `set` corresponds to the number in the file name following the word 'set'. This functionality is provided to increase computational efficiency when subsetting to a cohort of patients which is very large (millions). This can be a computationally expensive process as each flatfile being read in, must be cross matched with a large vector . The CPRD flatfiles are split up into groups which can be identified from their naming convention. Patients from set 1, will have their data in DrugIssue, Observation, etc, all with the same "set" suffix in the flatfile name. We can utilise this to speed up the process of subsetting the data from the flatfiles to only those with patids in subset.patid. Instead we subset to those with patids in subset_patids, and with the corresponding value of "set", which matches the suffix "set" in the CPRD flatfile file name. For example, patients in the Patient file which had suffix "set1", will have their medical data in the Observation file with suffix "set1". When subsetting the Observation file to those in subset_patids (our cohort), we only need to do so for patients who were also in the patient file with suffix "set1". If the cohort of patients for which you want to subset the data to is very small, the computational gains from this argument are minor and it can be ignored.

The function for reading in the .txt file will be chosen from a set of functions provided with rcprd, based on the filetype (`filetype`). `extract_txt_func` does not need to be specified unless wanting to manually define the function for doing this. This may be beneficial if wanting to change variable formats, or if the variables in the .txt files change in future releases of CPRD AURUM and rcprd has not been updated.

Value

Adds .txt file to SQLite database on hard disk.

Examples

```
## Create connection to a temporary database
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Add observation data from all observation files in specified directory
cprd_extract(db = aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation")

## Query database
RSQLite::dbGetQuery(aurum_extract, 'SELECT * FROM observation', n = 3)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

---

create_directory_system

Create the appropriate directory system to be able to run functions without specifying hard filepaths

---

Description

Create the appropriate directory system to be able to run functions without specifying hard filepaths

Usage

```
create_directory_system(rootdir = NULL)
```

Arguments

`rootdir`	Directory within which to create the directory system

Value

No return value, creates directory system in the specified directory.

Examples

```
## Create directory system compatible with rcprd's automatic saving of output
create_directory_system(tempdir())
file.exists(file.path(tempdir(),"data"))
file.exists(file.path(tempdir(),"code"))
file.exists(file.path(tempdir(),"codelists"))

## Return filespace to how it was prior to example
delete_directory_system(tempdir())
```

---

db_query *Query an RSQLite database.*

Description

Query an RSQLite database stored on the hard disk for observations with specific codes.

Usage

```
db_query(
  codelist,
  db_open = NULL,
  db = NULL,
  db_filepath = NULL,
  db_cprd = c("aurum", "gold"),
 tab = c("observation", "drugissue", "clinical", "immunisation", "test", "therapy",
    "hes_primary", "death"),
  codelist_vector = NULL
)
```

Arguments

codelist	Name of codelist to query the database with.
db_open	An open SQLite database connection created using RSQLite::dbConnect, to be queried.
db	Name of SQLITE database on hard disk, to be queried.
db_filepath	Full filepath to SQLITE database on hard disk, to be queried.
db_cprd	CPRD Aurum ('aurum') or gold ('gold').
tab	Name of table in SQLite database that is to be queried.
codelist_vector	Vector of codes to query the database with. This takes precedent over codelist if both are specified.

Details

Specifying db requires a specific underlying directory structure. The SQLite database must be stored in "data/sql/" relative to the working directory. If the SQLite database is accessed through db, the connection will be opened and then closed after the query is complete. The same is true if the database is accessed through db_filepath. A connection to the SQLite database can also be opened manually using RSQLite::dbConnect, and then using the object as input to parameter db_open. After wards, the connection must be closed manually using RSQLite::dbDisconnect. If db_open is specified, this will take precedence over db or db_filepath.

Specifying codelist requires a specific underlying directory structure. The codelist on the hard disk must be stored in "codelists/analysis/" relative to the working directory, must be a .csv file, and contain a column "medcodeid", "prodcodeid" or "ICD10" depending on the chosen tab. The codelist can also be read in manually, and supplied as a character vector to codelist_vector. If codelist_vector is defined, this will take precedence over codelist.

Value

A data.table with observations contained in the specified codelist.

Examples

```
## Create connection to a temporary database
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Add observation data from all observation files in specified directory
cprd_extract(db = aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation")

## Query database for a specific medcode
db_query(db_open = aurum_extract,
tab ="observation",
codelist_vector = "187341000000114")

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

---

delete_directory_system

Deletes directory system created by delete_directory_system

Description

Deletes directory system created by delete_directory_system. Primarily used to restore filespaces to original in examples/tests/vignettes.

Usage

```
delete_directory_system(rootdir = NULL)
```

Arguments

rootdir	Directory within which to delete the directory system

Value

No return value, deletes directory system in the specified directory.

Examples

```
## Print current working directory
getwd()

## Create directory system
create_directory_system(tempdir())
file.exists(file.path(tempdir(),"data"))
file.exists(file.path(tempdir(),"code"))
file.exists(file.path(tempdir(),"codelists"))

## Return filespace to how it was prior to example
delete_directory_system(tempdir())
file.exists(file.path(tempdir(),"data"))
file.exists(file.path(tempdir(),"code"))
file.exists(file.path(tempdir(),"codelists"))
```

---

extract_bmi *Extract most recent BMI score relative to an index date.*

Description

Extract most recent BMI score relative to an index date.

Usage

```
extract_bmi(
  cohort,
  varname = NULL,
  codelist_bmi = NULL,
  codelist_weight = NULL,
  codelist_height = NULL,
  codelist_bmi_vector = NULL,
  codelist_weight_vector = NULL,
  codelist_height_vector = NULL,
  indexdt,
```

```
  t = NULL,
  t_varname = TRUE,
  time_prev = 365.25 * 5,
  time_post = 0,
  lower_bound = -Inf,
  upper_bound = Inf,
  db_open = NULL,
  db = NULL,
  db_filepath = NULL,
  out_save_disk = FALSE,
  out_subdir = NULL,
  out_filepath = NULL,
  return_output = TRUE
)
```

Arguments

cohort	Cohort to extract age for.
varname	Optional name for variable in output dataset.
codelist_bmi	Name of codelist (stored on hard disk in "codelists/analysis/") for BMI to query the database with.
codelist_weight	Name of codelist (stored on hard disk in "codelists/analysis/") for weight to query the database with.
codelist_height	Name of codelist (stored on hard disk in "codelists/analysis/") for height to query the database with.
codelist_bmi_vector	Vector of codes for BMI to query the database with.
codelist_weight_vector	Vector of codes for weight to query the database with.
codelist_height_vector	Vector of codes for height to query the database with.
indexdt	Name of variable which defines index date in cohort.
t	Number of days after index date at which to calculate variable.
t_varname	Whether to add t to varname.
time_prev	Number of days prior to index date to look for codes.
time_post	Number of days after index date to look for codes.
lower_bound	Lower bound for returned values.
upper_bound	Upper bound for returned values.
db_open	An open SQLite database connection created using RSQLite::dbConnect, to be queried.
db	Name of SQLITE database on hard disk (stored in "data/sql/"), to be queried.
db_filepath	Full filepath to SQLITE database on hard disk, to be queried.

out_save_disk	If TRUE will attempt to save outputted data frame to directory "data/extraction/".
out_subdir	Sub-directory of "data/extraction/" to save outputted data frame into.
out_filepath	Full filepath and filename to save outputted data frame into.
return_output	If TRUE will return outputted data frame into R workspace.

Details

BMI can either be identified through a directly recorded BMI score, or calculated via height and weight scores. Full details on the algorithm for extracting BMI are given in the vignette: Details-on-algorithms-for-extracting-specific-variables. This vignette can be viewed by running `vignette("help", package = "rcprd")`.

Specifying db requires a specific underlying directory structure. The SQLite database must be stored in "data/sql/" relative to the working directory. If the SQLite database is accessed through db, the connection will be opened and then closed after the query is complete. The same is true if the database is accessed through db_filepath. A connection to the SQLite database can also be opened manually using `RSQLite::dbConnect`, and then using the object as input to parameter db_open. After wards, the connection must be closed manually using `RSQLite::dbDisconnect`. If db_open is specified, this will take precedence over db or db_filepath.

If `out_save_disk = TRUE`, the data frame will automatically be written to an .rds file in a subdirectory "data/extraction/" of the working directory. This directory structure must be created in advance. out_subdir can be used to specify subdirectories within "data/extraction/". These options will use a default naming convetion. This can be overwritten using out_filepath to manually specify the location on the hard disk to save. Alternatively, return the data frame into the R workspace using `return_output = TRUE` and then save onto the hard disk manually.

Specifying the non-vector type codelists requires a specific underlying directory structure. The codelist on the hard disk must be stored in "codelists/analysis/" relative to the working directory, must be a .csv file, and contain a column "medcodeid", "prodcodeid" or "ICD10" depending on the chosen tab. The input to these variables should just be the name of the files (excluding the suffix .csv). The codelists can also be read in manually, and supplied as a character vector. This option will take precedence over the codelists stored on the hard disk if both are specified.

Value

A data frame with variable BMI.

Examples

```
## Connect
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Create SQLite database using cprd_extract
cprd_extract(aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation", use_set = FALSE)

## Define cohort and add index date
pat<-extract_cohort(system.file("aurum_data", package = "rcprd"))
pat$indexdt <- as.Date("01/01/1955", format = "%d/%m/%Y")
```

```
## Extract most recent BMI prior to index date
extract_bmi(cohort = pat,
codelist_bmi_vector = "498521000006119",
codelist_weight_vector = "401539014",
codelist_height_vector = "13483031000006114",
indexdt = "indexdt",
time_prev = Inf,
db_open = aurum_extract,
return_output = TRUE)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

`extract_cholhdl_ratio` *Extract most recent total cholesterol/high-density lipoprotein ratio score relative to an index date.*

Description

Extract most recent total cholesterol/high-density lipoprotein ratio score relative to an index date.

Usage

```
extract_cholhdl_ratio(
  cohort,
  varname = NULL,
  codelist_ratio = NULL,
  codelist_chol = NULL,
  codelist_hdl = NULL,
  codelist_ratio_vector = NULL,
  codelist_chol_vector = NULL,
  codelist_hdl_vector = NULL,
  indexdt,
  t = NULL,
  t_varname = TRUE,
  time_prev = 365.25 * 5,
  time_post = 0,
  lower_bound = -Inf,
  upper_bound = Inf,
  db_open = NULL,
  db = NULL,
  db_filepath = NULL,
  out_save_disk = FALSE,
  out_subdir = NULL,
  out_filepath = NULL,
  return_output = TRUE
)
```

Arguments

cohort	Cohort to extract age for.
varname	Optional name for variable in output dataset.
codelist_ratio	Name of codelist (stored on hard disk in "codelists/analysis/") for ratio to query the database with.
codelist_chol	Name of codelist (stored on hard disk in "codelists/analysis/") for total cholesterol to query the database with.
codelist_hdl	Name of codelist (stored on hard disk in "codelists/analysis/") for high-density lipoprotein to query the database with.
codelist_ratio_vector	Vector of codes for ratio to query the database with.
codelist_chol_vector	Vector of codes for total cholesterol to query the database with.
codelist_hdl_vector	Vector of codes for high-density lipoprotein to query the database with.
indexdt	Name of variable which defines index date in cohort.
t	Number of days after index date at which to calculate variable.
t_varname	Whether to add t to varname.
time_prev	Number of days prior to index date to look for codes.
time_post	Number of days after index date to look for codes.
lower_bound	Lower bound for returned values.
upper_bound	Upper bound for returned values.
db_open	An open SQLite database connection created using RSQLite::dbConnect, to be queried.
db	Name of SQLITE database on hard disk (stored in "data/sql/"), to be queried.
db_filepath	Full filepath to SQLITE database on hard disk, to be queried.
out_save_disk	If TRUE will attempt to save outputted data frame to directory "data/extraction/".
out_subdir	Sub-directory of "data/extraction/" to save outputted data frame into.
out_filepath	Full filepath and filename to save outputted data frame into.
return_output	If TRUE will return outputted data frame into R workspace.

Details

Cholesterol/HDL ratio can either be identified through a directly recorded cholesterol/hdl ratio score, or calculated via total cholesterol and HDL scores. Full details on the algorithm for extracting cholesterol/hdl ratio are given in the vignette: Details-on-algorithms-for-extracting-specific-variables. This vignette can be viewed by running `vignette("help", package = "rcprd")`.

Specifying db requires a specific underlying directory structure. The SQLite database must be stored in "data/sql/" relative to the working directory. If the SQLite database is accessed through db, the connection will be opened and then closed after the query is complete. The same is true if the database is accessed through db_filepath. A connection to the SQLite database can also be opened manually using `RSQLite::dbConnect`, and then using the object as input to parameter

db_open. After wards, the connection must be closed manually using RSQLite::dbDisconnect. If db_open is specified, this will take precedence over db or db_filepath.

If out_save_disk = TRUE, the data frame will automatically be written to an .rds file in a subdirectory "data/extraction/" of the working directory. This directory structure must be created in advance. out_subdir can be used to specify subdirectories within "data/extraction/". These options will use a default naming convetion. This can be overwritten using out_filepath to manually specify the location on the hard disk to save. Alternatively, return the data frame into the R workspace using return_output = TRUE and then save onto the hard disk manually.

Specifying the non-vector type codelists requires a specific underlying directory structure. The codelist on the hard disk must be stored in "codelists/analysis/" relative to the working directory, must be a .csv file, and contain a column "medcodeid", "prodcodeid" or "ICD10" depending on the chosen tab. The input to these variables should just be the name of the files (excluding the suffix .csv). The codelists can also be read in manually, and supplied as a character vector. This option will take precedence over the codelists stored on the hard disk if both are specified.

Value

A data frame with variable total cholesterol/high-density lipoprotein ratio.

Examples

```
## Connect
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Create SQLite database using cprd_extract
cprd_extract(aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation", use_set = FALSE)

## Define cohort and add index date
pat<-extract_cohort(system.file("aurum_data", package = "rcprd"))
pat$indexdt <- as.Date("01/01/1955", format = "%d/%m/%Y")

## Extract most recent cholhdl_ratio prior to index date
extract_cholhdl_ratio(cohort = pat,
codelist_ratio_vector = "498521000006119",
codelist_chol_vector = "401539014",
codelist_hdl_vector = "13483031000006114",
indexdt = "indexdt",
time_prev = Inf,
db_open = aurum_extract,
return_output = TRUE)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

extract_cohort *Create cohort from patient files*

Description

Create cohort from patient files

Usage

```
extract_cohort(filepath, patids = NULL, select = NULL, set = FALSE)
```

Arguments

filepath	Path to directory containing .txt files.
patids	Patids of patients to retain in the cohort. Character vector. Numeric values should not be used.
select	Character vector of column names to select.
set	If TRUE will create a variable called set which will contain the number that comes after the word 'set' in the file name.

Value

Data frame with patient information

Examples

```
## Extract cohort data
pat<-extract_cohort(filepath = system.file("aurum_data", package = "rcprd"))
pat
```

extract_diabetes *Extract diabetes status prior to an index date.*

Description

Extract diabetes status prior to an index date.

Usage

```
extract_diabetes(
  cohort,
  varname = NULL,
  codelist_type1 = NULL,
  codelist_type2 = NULL,
  codelist_type1_vector = NULL,
  codelist_type2_vector = NULL,
  indexdt,
  t = NULL,
  t_varname = TRUE,
  db_open = NULL,
  db = NULL,
  db_filepath = NULL,
  out_save_disk = FALSE,
  out_subdir = NULL,
  out_filepath = NULL,
  return_output = TRUE
)
```

Arguments

cohort	Cohort to extract age for.
varname	Optional name for variable in output dataset.
codelist_type1	Name of codelist (stored on hard disk in "codelists/analysis/") for type 1 diabetes to query the database with.
codelist_type2	Name of codelist (stored on hard disk in "codelists/analysis/") for type 2 diabetes to query the database with.
codelist_type1_vector	Vector of codes for type 1 diabetes to query the database with.
codelist_type2_vector	Vector of codes for type 2 diabetes to query the database with.
indexdt	Name of variable which defines index date in cohort.
t	Number of days after index date at which to calculate variable.
t_varname	Whether to add t to varname.
db_open	An open SQLite database connection created using RSQLite::dbConnect, to be queried.
db	Name of SQLITE database on hard disk (stored in "data/sql/"), to be queried.
db_filepath	Full filepath to SQLITE database on hard disk, to be queried.
out_save_disk	If TRUE will attempt to save outputted data frame to directory "data/extraction/".
out_subdir	Sub-directory of "data/extraction/" to save outputted data frame into.
out_filepath	Full filepath and filename to save outputted data frame into.
return_output	If TRUE will return outputted data frame into R workspace.

Details

If an individual is found to have medical codes for both type 1 and type 2 diabetes, the returned value of diabetes status will be type 1 diabetes. Full details on the algorithm for extracting diabetes status are given in the vignette: Details-on-algorithms-for-extracting-specific-variables. This vignette can be viewed by running `vignette("help", package = "rcprd")`.

Specifying `db` requires a specific underlying directory structure. The SQLite database must be stored in "data/sql/" relative to the working directory. If the SQLite database is accessed through `db`, the connection will be opened and then closed after the query is complete. The same is true if the database is accessed through `db_filepath`. A connection to the SQLite database can also be opened manually using `RSQLite::dbConnect`, and then using the object as input to parameter `db_open`. After wards, the connection must be closed manually using `RSQLite::dbDisconnect`. If `db_open` is specified, this will take precedence over `db` or `db_filepath`.

If `out_save_disk = TRUE`, the data frame will automatically be written to an .rds file in a subdirectory "data/extraction/" of the working directory. This directory structure must be created in advance. `out_subdir` can be used to specify subdirectories within "data/extraction/". These options will use a default naming convetion. This can be overwritten using `out_filepath` to manually specify the location on the hard disk to save. Alternatively, return the data frame into the R workspace using `return_output = TRUE` and then save onto the hard disk manually.

Specifying the non-vector type codelists requires a specific underlying directory structure. The codelist on the hard disk must be stored in "codelists/analysis/" relative to the working directory, must be a .csv file, and contain a column "medcodeid", "prodcodeid" or "ICD10" depending on the chosen `tab`. The input to these variables should just be the name of the files (excluding the suffix .csv). The codelists can also be read in manually, and supplied as a character vector. This option will take precedence over the codelists stored on the hard disk if both are specified.

Value

A data frame with variable diabetes status.

Examples

```
## Connect
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Create SQLite database using cprd_extract
cprd_extract(aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation", use_set = FALSE)

## Define cohort and add index date
pat<-extract_cohort(system.file("aurum_data", package = "rcprd"))
pat$indexdt <- as.Date("01/01/1955", format = "%d/%m/%Y")

## Extract diabetes prior to index date
extract_diabetes(cohort = pat,
codelist_type1_vector = "498521000006119",
codelist_type2_vector = "401539014",
indexdt = "indexdt",
db_open = aurum_extract)
```

```
## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

`extract_ho` *Extract a 'history of' type variable*

Description

Query an RSQLite database and return a data frame with a 0/1 vector depending on whether each individual has at least one observation with relevant code between a specified time period.

Usage

```
extract_ho(
  cohort,
  varname = NULL,
  codelist = NULL,
  codelist_vector = NULL,
  indexdt,
  t = NULL,
  t_varname = TRUE,
  time_prev = Inf,
  time_post = 0,
  numobs = 1,
  db_open = NULL,
  db = NULL,
  db_filepath = NULL,
  tab = c("observation", "drugissue", "hes_primary", "death"),
  out_save_disk = FALSE,
  out_subdir = NULL,
  out_filepath = NULL,
  return_output = TRUE
)
```

Arguments

`cohort`	Cohort of individuals to extract the 'history of' variable for.
`varname`	Name of variable in the outputted data frame.
`codelist`	Name of codelist (stored on hard disk) to query the database with.
`codelist_vector`	Vector of codes to query the database with. This takes precedent over `codelist` if both are specified.
`indexdt`	Name of variable in `cohort` which specifies the index date. The extracted variable will be calculated relative to this.

t	Number of days after indexdt at which to extract the variable.
t_varname	Whether to alter the variable name in the outputted data frame to reflect t.
time_prev	Number of days prior to index date to look for codes.
time_post	Number of days after index date to look for codes.
numobs	Number of obesrvations required to return a value of 1.
db_open	An open SQLite database connection created using RSQLite::dbConnect, to be queried.
db	Name of SQLITE database on hard disk (stored in "data/sql/"), to be queried.
db_filepath	Full filepath to SQLITE database on hard disk, to be queried.
tab	Table name to query in SQLite database.
out_save_disk	If TRUE will attempt to save outputted data frame to directory "data/extraction/".
out_subdir	Sub-directory of "data/extraction/" to save outputted data frame into.
out_filepath	Full filepath and filename to save outputted data frame into.
return_output	If TRUE will return outputted data frame into R workspace.

Details

Specifying db requires a specific underlying directory structure. The SQLite database must be stored in "data/sql/" relative to the working directory. If the SQLite database is accessed through db, the connection will be opened and then closed after the query is complete. The same is true if the database is accessed through db_filepath. A connection to the SQLite database can also be opened manually using RSQLite::dbConnect, and then using the object as input to parameter db_open. After wards, the connection must be closed manually using RSQLite::dbDisconnect. If db_open is specified, this will take precedence over db or db_filepath.

If out_save_disk = TRUE, the data frame will automatically be written to an .rds file in a subdirectory "data/extraction/" of the working directory. This directory structure must be created in advance. out_subdir can be used to specify subdirectories within "data/extraction/". These options will use a default naming convention. This can be overwritten using out_filepath to manually specify the location on the hard disk to save. Alternatively, return the data frame into the R workspace using return_output = TRUE and then save onto the hard disk manually.

Codelists can be specified in two ways. The first is to read the codelist into R as a character vector and then specify through the argument codelist_vector. Codelists stored on the hard disk can also be referred to from the codelist argument, but require a specific underlying directory structure. The codelist on the hard disk must be stored in a directory called "codelists/analysis/" relative to the working directory. The codelist must be a .csv file, and contain a column "medcodeid", "prodcodeid" or "ICD10" depending on the input for argument tab. The input to argument codelist should just be a character string of the name of the files (excluding the suffix '.csv'). The codelist_vector option will take precedence over the codelist argument if both are specified.

Value

A data frame with a 0/1 vector and patid. 1 = presence of code within the specified time period.

Examples

```
## Connect
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Create SQLite database using cprd_extract
cprd_extract(aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation", use_set = FALSE)

## Define cohort and add index date
pat<-extract_cohort(system.file("aurum_data", package = "rcprd"))
pat$indexdt <- as.Date("01/01/1955", format = "%d/%m/%Y")

## Extract a history of type variable prior to index date
extract_ho(pat,
codelist_vector = "187341000000114",
indexdt = "fup_start",
db_open = aurum_extract,
tab = "observation",
return_output = TRUE)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

---

extract_smoking *Extract smoking status prior to index date.*

Description

Extract smoking status prior to index date.

Usage

```
extract_smoking(
  cohort,
  varname = NULL,
  codelist_non = NULL,
  codelist_ex = NULL,
  codelist_light = NULL,
  codelist_mod = NULL,
  codelist_heavy = NULL,
  codelist_non_vector = NULL,
  codelist_ex_vector = NULL,
  codelist_light_vector = NULL,
  codelist_mod_vector = NULL,
  codelist_heavy_vector = NULL,
```

```
  indexdt,
  t = NULL,
  t_varname = TRUE,
  db_open = NULL,
  db = NULL,
  db_filepath = NULL,
  out_save_disk = FALSE,
  out_subdir = NULL,
  out_filepath = NULL,
  return_output = TRUE
)
```

Arguments

cohort	Cohort to extract age for.
varname	Optional name for variable in output dataset.
codelist_non	Name of codelist (stored on hard disk in "codelists/analysis/") for non-smoker to query the database with.
codelist_ex	Name of codelist (stored on hard disk in "codelists/analysis/") for ex-smoker to query the database with.
codelist_light	Name of codelist (stored on hard disk in "codelists/analysis/") for light smoker to query the database with.
codelist_mod	Name of codelist (stored on hard disk in "codelists/analysis/") for moderate smoker to query the database with.
codelist_heavy	Name of codelist (stored on hard disk in "codelists/analysis/") for heavy smoker to query the database with.
codelist_non_vector	Vector of codes for non-smoker to query the database with.
codelist_ex_vector	Vector of codes for ex-smoker to query the database with.
codelist_light_vector	Vector of codes for light smoker to query the database with.
codelist_mod_vector	Vector of codes for moderate smoker to query the database with.
codelist_heavy_vector	Vector of codes for heavy smoker to query the database with.
indexdt	Name of variable which defines index date in cohort.
t	Number of days after index date at which to calculate variable.
t_varname	Whether to add t to varname.
db_open	An open SQLite database connection created using RSQLite::dbConnect, to be queried.
db	Name of SQLITE database on hard disk (stored in "data/sql/"), to be queried.
db_filepath	Full filepath to SQLITE database on hard disk, to be queried.
out_save_disk	If TRUE will attempt to save outputted data frame to directory "data/extraction/".

out_subdir	Sub-directory of "data/extraction/" to save outputted data frame into.
out_filepath	Full filepath and filename to save outputted data frame into.
return_output	If TRUE will return outputted data frame into R workspace.

Details

Returns the most recent value of smoking status. If the most recently recorded observation of smoking status is non-smoker, but the individual has a history of smoking identified through the medical record, the outputted value of smoking status will be ex-smoker. Full details on the algorithm for extracting smoking status are given in the vignette: Details-on-algorithms-for-extracting-specific-variables. This vignette can be viewed by running `vignette("help", package = "rcprd")`.

Specifying db requires a specific underlying directory structure. The SQLite database must be stored in "data/sql/" relative to the working directory. If the SQLite database is accessed through db, the connection will be opened and then closed after the query is complete. The same is true if the database is accessed through db_filepath. A connection to the SQLite database can also be opened manually using `RSQLite::dbConnect`, and then using the object as input to parameter db_open. After wards, the connection must be closed manually using `RSQLite::dbDisconnect`. If db_open is specified, this will take precedence over db or db_filepath.

If out_save_disk = TRUE, the data frame will automatically be written to an .rds file in a subdirectory "data/extraction/" of the working directory. This directory structure must be created in advance. out_subdir can be used to specify subdirectories within "data/extraction/". These options will use a default naming convetion. This can be overwritten using out_filepath to manually specify the location on the hard disk to save. Alternatively, return the data frame into the R workspace using return_output = TRUE and then save onto the hard disk manually.

Specifying the non-vector type codelists requires a specific underlying directory structure. The codelist on the hard disk must be stored in "codelists/analysis/" relative to the working directory, must be a .csv file, and contain a column "medcodeid", "prodcodeid" or "ICD10" depending on the chosen tab. The input to these variables should just be the name of the files (excluding the suffix .csv). The codelists can also be read in manually, and supplied as a character vector. This option will take precedence over the codelists stored on the hard disk if both are specified.

We take the most recent smoking status record. If an individuals most recent smoking status is a non-smoker, but they have a history of smoking prior to this, these individuals will be classed as ex-smokers.

Value

A data frame with variable smoking status.

Examples

```
## Connect
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Create SQLite database using cprd_extract
cprd_extract(aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation", use_set = FALSE)
```

```
## Define cohort and add index date
pat<-extract_cohort(system.file("aurum_data", package = "rcprd"))
pat$indexdt <- as.Date("01/01/1955", format = "%d/%m/%Y")

## Extract smoking status prior to index date
extract_smoking(cohort = pat,
codelist_non_vector = "498521000006119",
codelist_ex_vector = "401539014",
codelist_light_vector = "128011000000115",
codelist_mod_vector = "380389013",
codelist_heavy_vector = "13483031000006114",
indexdt = "indexdt",
db_open = aurum_extract)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

`extract_test_data` *Extract test data.*

Description

Query an RSQLite database and return a data frame containing the most recent test result that meets specified criteria.

Usage

```
extract_test_data(
  cohort,
  varname = NULL,
  codelist = NULL,
  codelist_vector = NULL,
  indexdt,
  t = NULL,
  t_varname = TRUE,
  time_prev = Inf,
  time_post = 0,
  lower_bound = -Inf,
  upper_bound = Inf,
  numobs = 1,
  keep_numunit = FALSE,
  db_open = NULL,
  db = NULL,
  db_filepath = NULL,
  out_save_disk = FALSE,
  out_subdir = NULL,
```

```
  out_filepath = NULL,
  return_output = FALSE
)
```

Arguments

cohort	Cohort of individuals to extract the 'history of' variable for.
varname	Name of variable in the outputted data frame.
codelist	Name of codelist (stored on hard disk) to query the database with.
codelist_vector	Vector of codes to query the database with. This takes precedent over codelist if both are specified.
indexdt	Name of variable in cohort which specifies the index date. The extracted variable will be calculated relative to this.
t	Number of days after indexdt at which to extract the variable.
t_varname	Whether to alter the variable name in the outputted data frame to reflect t.
time_prev	Number of days prior to index date to look for codes.
time_post	Number of days after index date to look for codes.
lower_bound	Lower bound for returned values.
upper_bound	Upper bound for returned values.
numobs	Number of test results to return. Will return most recent values that are in the valid time and bound ranges.
keep_numunit	TRUE/FALSE whether to keep numunitid, medcodeid and obsdate in the outputted dataset
db_open	An open SQLite database connection created using RSQLite::dbConnect, to be queried.
db	Name of SQLITE database on hard disk (stored in "data/sql/"), to be queried.
db_filepath	Full filepath to SQLITE database on hard disk, to be queried.
out_save_disk	If TRUE will attempt to save outputted data frame to directory "data/extraction/".
out_subdir	Sub-directory of "data/extraction/" to save outputted data frame into.
out_filepath	Full filepath and filename to save outputted data frame into.
return_output	If TRUE will return outputted data frame into R workspace.

Details

Specifying db requires a specific underlying directory structure. The SQLite database must be stored in "data/sql/" relative to the working directory. If the SQLite database is accessed through db, the connection will be opened and then closed after the query is complete. The same is true if the database is accessed through db_filepath. A connection to the SQLite database can also be opened manually using RSQLite::dbConnect, and then using the object as input to parameter db_open. After wards, the connection must be closed manually using RSQLite::dbDisconnect. If db_open is specified, this will take precedence over db or db_filepath.

If out_save_disk = TRUE, the data frame will automatically be written to an .rds file in a subdirectory "data/extraction/" of the working directory. This directory structure must be created in advance. out_subdir can be used to specify subdirectories within "data/extraction/". These options will use a default naming convetion. This can be overwritten using out_filepath to manually specify the location on the hard disk to save. Alternatively, return the data frame into the R workspace using return_output = TRUE and then save onto the hard disk manually.

Codelists can be specified in two ways. The first is to read the codelist into R as a character vector and then specify through the argument codelist_vector. Codelists stored on the hard disk can also be referred to from the codelist argument, but require a specific underlying directory structure. The codelist on the hard disk must be stored in a directory called "codelists/analysis/" relative to the working directory. The codelist must be a .csv file, and contain a column "medcodeid", "prodcodeid" or "ICD10" depending on the input for argument tab. The input to argument codelist should just be a character string of the name of the files (excluding the suffix '.csv'). The codelist_vector option will take precedence over the codelist argument if both are specified.

Currently only returns most recent test result. This will be updated to return more than one most recent test result if specified.

Value

A data frame containing all test results that meets required criteria.

Examples

```
## Connect
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Create SQLite database using cprd_extract
cprd_extract(aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation", use_set = FALSE)

## Define cohort and add index date
pat<-extract_cohort(system.file("aurum_data", package = "rcprd"))
pat$indexdt <- as.Date("01/01/1955", format = "%d/%m/%Y")

## Extract most recent test value prior to index date
extract_test_data(pat,
codelist_vector = "187341000000114",
indexdt = "fup_start",
db_open = aurum_extract,
time_prev = Inf,
return_output = TRUE)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

`extract_test_data_var` *Extract standard deviation of all test data values over a specified time period relative to an index date.*

Description

Extract standard deviation of all test data values over a specified time period relative to an index date.

Usage

```
extract_test_data_var(
  cohort,
  varname = NULL,
  codelist,
  codelist_vector,
  indexdt,
  t = NULL,
  t_varname = TRUE,
  time_prev = 365.25 * 5,
  time_post = 0,
  lower_bound = -Inf,
  upper_bound = Inf,
  db_open = NULL,
  db = NULL,
  db_filepath = NULL,
  out_save_disk = FALSE,
  out_subdir = NULL,
  out_filepath = NULL,
  return_output = FALSE
)
```

Arguments

`cohort`	Cohort of individuals to extract the 'history of' variable for.
`varname`	Name of variable in the outputted data frame.
`codelist`	Name of codelist (stored on hard disk) to query the database with.
`codelist_vector`	Vector of codes to query the database with. This takes precedent over `codelist` if both are specified.
`indexdt`	Name of variable in `cohort` which specifies the index date. The extracted variable will be calculated relative to this.
`t`	Number of days after `indexdt` at which to extract the variable.
`t_varname`	Whether to alter the variable name in the outputted data frame to reflect `t`.
`time_prev`	Number of days prior to index date to look for codes.

time_post	Number of days after index date to look for codes.
lower_bound	Lower bound for returned values.
upper_bound	Upper bound for returned values.
db_open	An open SQLite database connection created using RSQLite::dbConnect, to be queried.
db	Name of SQLITE database on hard disk (stored in "data/sql/"), to be queried.
db_filepath	Full filepath to SQLITE database on hard disk, to be queried.
out_save_disk	If TRUE will attempt to save outputted data frame to directory "data/extraction/".
out_subdir	Sub-directory of "data/extraction/" to save outputted data frame into.
out_filepath	Full filepath and filename to save outputted data frame into.
return_output	If TRUE will return outputted data frame into R workspace.

Details

Specifying db requires a specific underlying directory structure. The SQLite database must be stored in "data/sql/" relative to the working directory. If the SQLite database is accessed through db, the connection will be opened and then closed after the query is complete. The same is true if the database is accessed through db_filepath. A connection to the SQLite database can also be opened manually using RSQLite::dbConnect, and then using the object as input to parameter db_open. After wards, the connection must be closed manually using RSQLite::dbDisconnect. If db_open is specified, this will take precedence over db or db_filepath.

If out_save_disk = TRUE, the data frame will automatically be written to an .rds file in a subdirectory "data/extraction/" of the working directory. This directory structure must be created in advance. out_subdir can be used to specify subdirectories within "data/extraction/". These options will use a default naming convetion. This can be overwritten using out_filepath to manually specify the location on the hard disk to save. Alternatively, return the data frame into the R workspace using return_output = TRUE and then save onto the hard disk manually.

Currently only returns most recent test result. This will be updated to return more than one most recent test result if specified.

Value

A data frame containing standard deviation of test results.

Examples

```
## Connect
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Create SQLite database using cprd_extract
cprd_extract(aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation", use_set = FALSE)

## Define cohort and add index date
pat<-extract_cohort(system.file("aurum_data", package = "rcprd"))
pat$indexdt <- as.Date("01/01/1955", format = "%d/%m/%Y")
```

```
## Extract standard deviation of previous test scores prior to index date
extract_test_data_var(pat,
codelist_vector = "187341000000114",
indexdt = "fup_start",
db_open = aurum_extract,
time_prev = Inf,
return_output = TRUE)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

`extract_test_recent` *Extract test data.*

Description

Query an RSQLite database and return a data frame containing the most recent test result that meets specified criteria.

Usage

```
extract_test_recent(
  cohort,
  varname = NULL,
  codelist = NULL,
  codelist_vector = NULL,
  indexdt,
  t = NULL,
  t_varname = TRUE,
  time_prev = 365.25 * 5,
  time_post = 0,
  lower_bound = -Inf,
  upper_bound = Inf,
  db_open = NULL,
  db = NULL,
  db_filepath = NULL,
  out_save_disk = FALSE,
  out_subdir = NULL,
  out_filepath = NULL,
  return_output = FALSE
)
```

Arguments

`cohort`	Cohort of individuals to extract the 'history of' variable for.

varname	Name of variable in the outputted data frame.
codelist	Name of codelist (stored on hard disk) to query the database with.
codelist_vector	Vector of codes to query the database with. This takes precedent over codelist if both are specified.
indexdt	Name of variable in cohort which specifies the index date. The extracted variable will be calculated relative to this.
t	Number of days after indexdt at which to extract the variable.
t_varname	Whether to alter the variable name in the outputted data frame to reflect t.
time_prev	Number of days prior to index date to look for codes.
time_post	Number of days after index date to look for codes.
lower_bound	Lower bound for returned values.
upper_bound	Upper bound for returned values.
db_open	An open SQLite database connection created using RSQLite::dbConnect, to be queried.
db	Name of SQLITE database on hard disk (stored in "data/sql/"), to be queried.
db_filepath	Full filepath to SQLITE database on hard disk, to be queried.
out_save_disk	If TRUE will attempt to save outputted data frame to directory "data/extraction/".
out_subdir	Sub-directory of "data/extraction/" to save outputted data frame into.
out_filepath	Full filepath and filename to save outputted data frame into.
return_output	If TRUE will return outputted data frame into R workspace.

Details

Specifying db requires a specific underlying directory structure. The SQLite database must be stored in "data/sql/" relative to the working directory. If the SQLite database is accessed through db, the connection will be opened and then closed after the query is complete. The same is true if the database is accessed through db_filepath. A connection to the SQLite database can also be opened manually using RSQLite::dbConnect, and then using the object as input to parameter db_open. After wards, the connection must be closed manually using RSQLite::dbDisconnect. If db_open is specified, this will take precedence over db or db_filepath.

If out_save_disk = TRUE, the data frame will automatically be written to an .rds file in a subdirectory "data/extraction/" of the working directory. This directory structure must be created in advance. out_subdir can be used to specify subdirectories within "data/extraction/". These options will use a default naming convetion. This can be overwritten using out_filepath to manually specify the location on the hard disk to save. Alternatively, return the data frame into the R workspace using return_output = TRUE and then save onto the hard disk manually.

Codelists can be specified in two ways. The first is to read the codelist into R as a character vector and then specify through the argument codelist_vector. Codelists stored on the hard disk can also be referred to from the codelist argument, but require a specific underlying directory structure. The codelist on the hard disk must be stored in a directory called "codelists/analysis/" relative to the working directory. The codelist must be a .csv file, and contain a column "medcodeid", "prodcodeid" or "ICD10" depending on the input for argument tab. The input to argument

codelist should just be a character string of the name of the files (excluding the suffix '.csv'). The codelist_vector option will take precedence over the codelist argument if both are specified.

Currently only returns most recent test result. This will be updated to return more than one most recent test result if specified.

Value

A data frame containing most recent test result that meets required criteria.

Examples

```
## Connect
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Create SQLite database using cprd_extract
cprd_extract(aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation", use_set = FALSE)

## Define cohort and add index date
pat<-extract_cohort(system.file("aurum_data", package = "rcprd"))
pat$indexdt <- as.Date("01/01/1955", format = "%d/%m/%Y")

## Extract most recent test value prior to index date
extract_test_data(pat,
codelist_vector = "187341000000114",
indexdt = "fup_start",
db_open = aurum_extract,
time_prev = Inf,
return_output = TRUE)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

---

extract_time_until *Extract a 'time until' type variable*

Description

Query an RSQLite database and a data frame with the time until first code of interest or censoring, and an event/censoring indicator.

Usage

```
extract_time_until(
  cohort,
  varname_time = NULL,
```

```
  varname_indicator = NULL,
  codelist = NULL,
  codelist_vector = NULL,
  indexdt,
  censdt,
  censdt_lag = 0,
  t = NULL,
  t_varname = TRUE,
  db_open = NULL,
  db = NULL,
  db_filepath = NULL,
  tab = c("observation", "drugissue", "hes_primary", "death"),
  out_save_disk = FALSE,
  out_subdir = NULL,
  out_filepath = NULL,
  return_output = FALSE
)
```

Arguments

cohort	Cohort of individuals to extract the variable for.
varname_time	Name of time variable in the outputted data frame.
varname_indicator	Name of event/censoring indicator in the outputted data frame.
codelist	Name of codelist (stored on hard disk) to query the database with.
codelist_vector	Vector of codes to query the database with. This takes precedent over codelist if both are specified.
indexdt	Name of variable in cohort which specifies the index date. The extracted variable will be calculated relative to this.
censdt	Name of variable in cohort which specifies the censoring date.
censdt_lag	Number of days after censoring where events will still be considered, to account for delays in recording.
t	Number of days after indexdt at which to extract the variable.
t_varname	Whether to alter the variable name in the outputted data frame to reflect t.
db_open	An open SQLite database connection created using RSQLite::dbConnect, to be queried.
db	Name of SQLITE database on hard disk (stored in "data/sql/"), to be queried.
db_filepath	Full filepath to SQLITE database on hard disk, to be queried.
tab	Table name to query in SQLite database.
out_save_disk	If TRUE will attempt to save outputted data frame to directory "data/extraction/".
out_subdir	Sub-directory of "data/extraction/" to save outputted data frame into.
out_filepath	Full filepath and filename to save outputted data frame into.
return_output	If TRUE will return outputted data frame into R workspace.

Details

Specifying db requires a specific underlying directory structure. The SQLite database must be stored in "data/sql/" relative to the working directory. If the SQLite database is accessed through db, the connection will be opened and then closed after the query is complete. The same is true if the database is accessed through db_filepath. A connection to the SQLite database can also be opened manually using RSQLite::dbConnect, and then using the object as input to parameter db_open. After wards, the connection must be closed manually using RSQLite::dbDisconnect. If db_open is specified, this will take precedence over db or db_filepath.

If out_save_disk = TRUE, the data frame will automatically be written to an .rds file in a subdirectory "data/extraction/" of the working directory. This directory structure must be created in advance. out_subdir can be used to specify subdirectories within "data/extraction/". These options will use a default naming convetion. This can be overwritten using out_filepath to manually specify the location on the hard disk to save. Alternatively, return the data frame into the R workspace using return_output = TRUE and then save onto the hard disk manually.

Codelists can be specified in two ways. The first is to read the codelist into R as a character vector and then specify through the argument codelist_vector. Codelists stored on the hard disk can also be referred to from the codelist argument, but require a specific underlying directory structure. The codelist on the hard disk must be stored in a directory called "codelists/analysis/" relative to the working directory. The codelist must be a .csv file, and contain a column "medcodeid", "prodcodeid" or "ICD10" depending on the input for argument tab. The input to argument codelist should just be a character string of the name of the files (excluding the suffix '.csv'). The codelist_vector option will take precedence over the codelist argument if both are specified.

If the time until event is the same as time until censored, this will be considered an event (var_indicator = 1)

If dtcens.lag > 0, then the time until the event of interest will be the time until the minimum of the event of interest, and date of censoring.

Value

A data frame with variable patid, a variable containing the time until event/censoring, and a variable containing event/censoring indicator.

Examples

```
## Connect
aurum_extract <- connect_database(file.path(tempdir(), "temp.sqlite"))

## Create SQLite database using cprd_extract
cprd_extract(aurum_extract,
filepath = system.file("aurum_data", package = "rcprd"),
filetype = "observation", use_set = FALSE)

## Define cohort and add index date and censoring date
pat<-extract_cohort(system.file("aurum_data", package = "rcprd"))
pat$indexdt <- as.Date("01/01/1955", format = "%d/%m/%Y")
pat$fup_end <- as.Date("01/01/2000", format = "%d/%m/%Y")

## Extract time until event/censoring
```

```
extract_time_until(pat,
codelist_vector = "187341000000114",
indexdt = "fup_start",
censdt = "fup_end",
db_open = aurum_extract,
tab = "observation",
return_output = TRUE)

## clean up
RSQLite::dbDisconnect(aurum_extract)
unlink(file.path(tempdir(), "temp.sqlite"))
```

`extract_txt_char` *Read in txt file with all colClasses = "character"*

Description

Read in txt file with all colClasses = "character"

Usage

```
extract_txt_char(filepath, ..., select = NULL)
```

Arguments

`filepath`	File path to raw .txt file
`...`	Arguments to pass onto utils::read.table
`select`	Character vector of variable names to select

`extract_txt_cons` *Read in raw .txt consultation file*

Description

Read in raw .txt consultation file

Usage

```
extract_txt_cons(filepath, ..., select = NULL)
```

Arguments

`filepath`	File path to raw .txt file
`...`	Arguments to pass onto utils::read.table
`select`	Character vector of variable names to select

extract_txt_death *Read in raw ONS death data file*

Description

Read in raw ONS death data file

Usage

```
extract_txt_death(filepath, ..., select = NULL)
```

Arguments

filepath	File path to raw .txt file
...	Arguments to pass onto utils::read.table
select	Character vector of variable names to select

extract_txt_drug *Read in raw .txt drugissue file*

Description

Read in raw .txt drugissue file

Usage

```
extract_txt_drug(filepath, ..., select = NULL)
```

Arguments

filepath	File path to raw .txt file
...	Arguments to pass onto utils::read.table
select	Character vector of variable names to select

extract_txt_hes_primary *Read in raw HES primary diagnoses file*

Description

Read in raw HES primary diagnoses file

Usage

```
extract_txt_hes_primary(filepath, ..., select = NULL)
```

Arguments

filepath	File path to raw .txt file
...	Arguments to pass onto utils::read.table
select	Character vector of variable names to select

extract_txt_linkage *Read in linkage eligibility file*

Description

Read in linkage eligibility file

Usage

```
extract_txt_linkage(filepath, ...)
```

Arguments

filepath	File path to raw .txt file
...	Arguments to pass onto utils::read.table

extract_txt_obs *Read in raw .txt observation file*

Description

Read in raw .txt observation file

Usage

```
extract_txt_obs(filepath, ..., select = NULL)
```

Arguments

filepath	File path to raw .txt file
...	Arguments to pass onto utils::read.table
select	Character vector of variable names to select

extract_txt_pat *Read in raw .txt patient file*

Description

Read in raw .txt patient file

Usage

```
extract_txt_pat(filepath, ..., set = FALSE)
```

Arguments

filepath	File path to raw .txt file
...	Arguments to pass onto utils::read.table
set	If TRUE will create a variable called set which will contain the number that comes after the word 'set' in the file name.

extract_txt_prob *Read in raw .txt problem file*

Description

Read in raw .txt problem file

Usage

```
extract_txt_prob(filepath, ..., select = NULL)
```

Arguments

filepath	File path to raw .txt file
...	Arguments to pass onto utils::read.table
select	Character vector of variable names to select

extract_txt_ref *Read in raw .txt referral file*

Description

Read in raw .txt referral file

Usage

```
extract_txt_ref(filepath, ..., select = NULL)
```

Arguments

filepath	File path to raw .txt file
...	Arguments to pass onto utils::read.table
select	Character vector of variable names to select

`implement_output` *Internal function to implement saving extracted variable to disk or returning into R workspace.*

Description

Will save extracted variable to disk if `out_save_disk = TRUE`. Note it relies on correct underlying structure of directories. Will output extracted variable into R workspace if `return_output = TRUE`.

Usage

```
implement_output(
  variable_dat,
  varname,
  out_save_disk,
  out_subdir,
  out_filepath,
  return_output
)
```

Arguments

`variable_dat`	Dataset containing variable
`varname`	Name of variable to use in filename
`out_save_disk`	If TRUE will save output to disk
`out_subdir`	Sub-directory of data/ to save output into
`out_filepath`	Full fiilepath to save dat onto
`return_output`	If TRUE returns output into R workspace

Index