Package: sendigR (via r-universe)

June 6, 2026

Title Enable Cross-Study Analysis of 'CDISC' 'SEND' Datasets

Version 1.0.2

Description A system enables cross study Analysis by extracting and filtering study data for control animals from 'CDISC' 'SEND' Study Repository. These data types are supported: Body Weights, Laboratory test results and Microscopic findings. These database types are supported: 'SQLite' and 'Oracle'.

License MIT + file LICENSE

URL https://github.com/phuse-org/sendigR

BugReports https://github.com/phuse-org/sendigR/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports DBI, RPostgres, RSQLite, data.table, readxl, magrittr, xfun, stringr, DescTools, parsedate, shiny, shinydashboard, htmltools, DT, dplyr, ggplot2, Hmisc, haven, plotly, cicerone, reticulate, sjlabelled

Suggests knitr, rmarkdown, logr, shinycssloaders, testthat

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 4.1.0)

SystemRequirements Python(>=3.9.6)

Config/pak/sysreqs cmake make libicu-dev libpng-dev libuv1-dev libssl-dev libpq-dev python3 libx11-dev zlib1g-dev

Repository https://yousuf28.r-universe.dev

Date/Publication 2025-03-06 19:03:41 UTC

RemoteUrl https://github.com/phuse-org/sendigR

RemoteRef HEAD

RemoteSha beb4a17066c4ec34e48f4c6aaced346d11111477

Contents

dbCreateIndexes . . . 2
dbCreateSchema . . . 3
dbDeleteStudies . . . 4
dbImportOneStudy . . . 5
dbImportStudies . . . 7
disconnectDB . . . 9
execSendDashboard . . . 9
gen_vocab . . . 10
genericQuery . . . 11
getControlSubj . . . 11
getFindingsPhase . . . 14
getFindingsSubjAge . . . 17
getStudiesSDESIGN . . . 20
getStudiesSTSTDTC . . . 22
getSubjData . . . 25
getSubjRoute . . . 26
getSubjSex . . . 29
getSubjSpeciesStrain . . . 31
getTabColLabels . . . 34
initEnvironment . . . 34
standardize_file . . . 36

Index **38**

`dbCreateIndexes` *Create indexes in SEND database*

Description

Create a set of indexes on the tables in a SEND database to optimize performance of extraction of data from the different functions in the package.

Usage

```
dbCreateIndexes(dbToken, replaceExisting = FALSE)
```

Arguments

`dbToken`	Mandatory Token for the open database connection (see `initEnvironment`).
`replaceExisting`	Mandatory, character Whether an already existing set of indexes in the database may be replaced by a new set of indexes.

Details

All the indexes are named <domain name>_sendigr_<nn> - .e.g. TS_sendigr_01.
If any additional indexes are manually created in the database, avoid to include 'sendigr' in the name, because all existing indexes with that included in the name will be initially deleted when execution the function with replaceExisting = TRUE.
It's recommended to wait with the creation of the indexes until the major amount of studies to be loaded in to the database are loaded.

Databases supported are SQLite and PostgreSQL, the checkDbType function makes sure one of those types are used

Value

No return value, called for side effects

Examples

```
## Not run:
createAllIndexes(myDbToken)

## End(Not run)
```

---

dbCreateSchema *Create a SEND schema in an open and empty database*

Description

Create all the domains and variables which are described in the SEND IG versions 3.0 and 3.1 in the database - i.e. a union of domains from the SEND IG versions and in each domain a union of variables from the SEND IG versions.

Usage

```
dbCreateSchema(dbToken)
```

Arguments

dbToken	Mandatory Token for the open database connection (see initEnvironment).

Details

Databases supported are SQLite and PostgreSQL, the checkDbType function makes sure one of those types are used

Value

No return value, called for side effects

Examples

```
## Not run:
# Create an empty SQLite database and create the SEND schema
myDbToken <- initEnvironment(dbType = 'sqlite',
                             dbPath ='/mydatapath/db/send.db',
                             dbCreate = TRUE)
dbCreateSchema(myDbToken)

## End(Not run)
```

`dbDeleteStudies` *Delete one or more studies in SEND database*

Description

Deletes data from all domains for one or more studies in a SEND database

Usage

```
dbDeleteStudies(dbToken, studyIdList)
```

Arguments

`dbToken`	Mandatory Token for the open database connection (see `initEnvironment`).
`studyIdList`	Mandatory, character A list or vector of study id values

Details

Databases supported are SQLite and PostgreSQL, the checkDbType function makes sure one of those types are used

Value

No return value, called for side effects

Examples

```
## Not run:
# delete one study
dbDeleteStudies(myDbToken, '122312')
# delete multiple studies
dbDeleteStudies(myDbToken, list('122312', '552343', '0942347'))

## End(Not run)
```

`dbImportOneStudy` *Import SEND study data in SAS xport format into a SEND database from a single study folder*

Description

Check each of the SAS xpt file located in the specified folder - import content from file and load it into the corresponding SEND domain table in the open database.

Usage

```
dbImportOneStudy(dbToken, xptPath, overWrite = FALSE, checkRequiredVars = TRUE)
```

Arguments

`dbToken`	Mandatory Token for the open database connection (see `initEnvironment`).
`xptPath`	Mandatory, character Location of the SAS xport files
`overWrite`	Mandatory, boolean Whether an already existing study in the database may be overwritten by newly imported data.
`checkRequiredVars`	Mandatory, boolean Whether not-required domains are checked for existence and content of required variables

Details

These requirements to the content of the folder must be fulfilled:

1. The folder must contain some SAS xport files named `[send domain].xpt` - the case of the file names doesn't care
2. A minimum set of required domain files must be included: `ts.xpt`, `tx.xpt`, `dm.xpt`.
3. Each xpt file must contain one data table with same name as the file name - i.e. a send domain name.

4. Each xpt file must contain a non-empty STUDYID value in each row equal to the value of TS.STUDYID.
5. Each xpt file must contain a set of required column(s).
   In general it's (where relevant for the different kinds of domains):
   `STUDYID`, `DOMAIN`, `--SEQ`, `USUBJID`, `--TESTCD`, `--TEST`,`--ORRES`, `--ORRESU`, `--STRESC`, `--STRESN`, `--STRESU`
6. The DOMAIN variable must contain the name of the actual domain in all rows

The last two requirements are checked for the required domains in all cases. For other domains, these two requirements are only checked if parameter `checkRequiredVars = TRUE`.

If an error is detected, the import and load of data is canceled, and further execution is aborted (i.e. error message is written to the console).
These error situations are checked and reported:

- Any of the requirements 1 to 3 are not fulfilled or any of the following requirements are not fulfilled for one of the required domains
- A study with the same value if STUDYID exists in the database and parameter `overWrite = FALSE`.

If one of the requirements 4 to 6 are not fulfilled for a not-required domain, this domain is excluded from the import. These kinds of issues are reported as one warning message to the console when data has been loaded.

Some non-critical issues, which doesn't prohibit data to be loaded to the database may be detected. These are reported as one warning message to the console when data has been loaded (together with eventual warning messages for skipped domains).
These non-critical issues are checked and reported:

- The study folder contains one or more xpt file(s) with names(s) not matching SEND domain name(s).
  Such files are ignored by the import/load process.
- An imported data tables contains one or more column(s) which do(es)n't exist(s) in the corresponding domain.

Databases supported are SQLite and PostgreSQL, the checkDbType function makes sure one of those types are used

Value

No return value, called for side effects

Examples

```
## Not run:
# Do not overwrite if study already exists in the database
dbImportOneStudy(myDbToken,'/mydatapath/studies/1213443')
# Allow to overwrite data if study already exists in the database
```

```
dbImportOneStudy(myDbToken,'/mydatapath/studies/786756', overwrite = TRUE)

## End(Not run)
```

`dbImportStudies` *Import SEND study data in SAS xport format into a SEND database from a hierarchy study folders.*

Description

For each non-empty folder below the specified root folder, the actions to import a set of SAS xpt files into the opened SQLlite database described for function dbImportOneStudy.

Usage

```
dbImportStudies(
  dbToken,
  xptPathRoot,
  overWrite = FALSE,
  checkRequiredVars = TRUE,
  verbose = FALSE,
  logFilePath = NULL
)
```

Arguments

`dbToken`	Mandatory Token for the open database connection (see `initEnvironment`).
`xptPathRoot`	Mandatory, character Root location of a set of sub folders - each sub folder with a set of SAS xport files for one study to import. The folder tree is traversed recursively - i.e. a multilevel folder hierarchy is allowed.
`overWrite`	Mandatory, boolean Whether an already existing study in the database may be overwritten by newly imported data.
`checkRequiredVars`	Mandatory, boolean Whether not-required domains are checked for existence and content of required variables
`verbose`	Mandatory, boolean Whether the status of the import shall be continuously written to the console for for each processed sub folder.

logFilePath — Optional, character
A path to a folder to contain a log file with the status of the import for each processed sub folder.
The name of the log file is logFilePath/dbImportStudies_<date & time>.log where <date & time> is the actual date and time in format YYYYmmdd_HH24MISS - e.g. dbImportStudies_20210323_084150.log if the function was called 23. March 2021 at 8:41:50

Details

The status for the processing of each sub folder is caught and returned as described below.
If parameter verbose = TRUE, the status for each processed sub folder is also printed to the console each time a sub folder has been processed - i.e. it's possible to followed the progress of the import process. If parameter logFilePath has been specified with an existing path to a folder, the status for each processed sub folder is also printed to a log file in this folder each time a sub folder has been processed.

Databases supported are SQLite and PostgreSQL, the checkDbType function makes sure one of those types are used

Value

A list containing a named element with the import status for each of the processed sub folders.
Each of the statuses are one of three variants:

- 'OK' - the SAS xport files has been imported to the database with no errors or warnings
- 'Warning: [list of warnings]' - the SAS xport files has been imported to the database but have one or more warnings
- 'Cancelled: [error message]' - the SAS xport files have not been imported to the database because an error has been detected.

Examples

```
## Not run:
# Import studies from a set of folders - do not allow to overwrite
# existing study data in the database, follow the progress
dbImportStudies(myDbToken,'/mydatapath/studies', verbose = TRUE)
# Import studies from another set of folders - allow to overwrite existing
# study data in the database
dbImportStudies(myDbToken,'/mydatapath/project123/studies', overwrite = TRUE)
# Import studies from a set of folders , save the status of each study load
# in a log file
dbImportStudies(myDbToken,'/mydatapath/studies',
                logFilePath = '/my/log file/path')

## End(Not run)
```

disconnectDB *Disconnect from the open database.*

Description

Close database session and disconnect from open database.

Usage

```
disconnectDB(dbToken)
```

Arguments

dbToken	Mandatory Token for the open database connection (see initEnvironment).

Value

No return value, called for side effects

Examples

```
## Not run:
disconnectDB()

## End(Not run)
```

execSendDashboard *Execute sendDashboard app*

Description

Executes an encapsulated Shiny which to query, visualize and extract historical control data from a SEND database.

Usage

```
execSendDashboard(dbToken)
```

Arguments

dbToken	Mandatory - token for the open database connection

Value

The function does not return anything, but it is possible to extract data from the app in different formats to use for further processing

Examples

```
## Not run:
dbToken <- initEnvironment(dbType='sqlite', dbPath='/path/to/database/send.db')
execSendDashboard(dbToken)
disconnectDB(dbToken)

## End(Not run)
```

---

gen_vocab *Create json file for vocabulary mappings. Keys are synonyms and values are the CDISC Controlled Terminology Submission values. Vocabularies are defined by column values from the tab-delimited files.*

Description

Create json file for vocabulary mappings. Keys are synonyms and values are the CDISC Controlled Terminology Submission values. Vocabularies are defined by column values from the tab-delimited files.

Usage

```
gen_vocab(in_file, out_path)
```

Arguments

in_file	Mandatory. List of tab-delimited files with synonyms and preferred terms.
out_path	Mandatory. output json filename.

Value

No return value, called for side effects

Examples

```
## Not run:
gen_vocab(list(infile1, infile2),jsonfile)

## End(Not run)
```

genericQuery *Execute database query and returns fetched rows.*

Description

The function executes a SQL select statements in the database and returns the fetched set of rows as a data.table.

Usage

```
genericQuery(dbToken, queryString, queryParams = NULL)
```

Arguments

dbToken	Mandatory Token for the open database connection (see initEnvironment).
queryString	Mandatory, character. The select statement to execute
queryParams	Optional, character. A variable with values for bind variable referenced in the where clause of the select statement

Value

Data.table with the set of fetched rows

Examples

```
## Not run:
genericQuery(dbToken,
             'select studyid, tsseq, tsgrpid, tsparmcd, tsval from ts')
genericQuery(dbToken,
           'select studyid, tsval from ts where tsparmcd = "SDESIGN" and studyid in (:1)',
             list("1234546","222333","444555"))

## End(Not run)
```

getControlSubj *Extract a list of control animals for a list of studies*

Description

Returns a data table with a list of animals belonging to the groups for negative control in the given list of studies.

Usage

```
getControlSubj(dbToken, studyList, inclUncertain = FALSE)
```

Arguments

dbToken	Mandatory. Token for the open database connection (see initEnvironment).
studyList	Mandatory, data.table. A table with a list of studies to limit the output to be within this set of studies. The table must include a column named 'STUDYID'.
inclUncertain	Mandatory, boolean. Indicates whether animals, which cannot be identified as neither negative nor positive control (i.e. uncertain animals), shall be included or not in the output data table.

Details

The set of animals contains all animals from DM where the SETCD is associated with a TX parameter 'TCNTRL'. Negative control animals are further defined by

- either containing a word from a set of words, to automatically distinguish it as a negative control:
  - ['placebo', 'untreated', 'sham']
- or containing a combination of a word from of two lists:
  1. ['negative', 'saline', 'peg', 'vehicle', 'citrate', 'dextrose', 'water', 'air']
  2. ['item', 'control', 'article']

Animals are in all cases excluded (i.e. whether `inclUncertain=TRUE` or `inclUncertain=FALSE`) from the output set, when they are identified as positive control animals - i.e. they are associated with a TX parameter 'TCNTRL' containing a word from this set of words:

- ['positive','reference']

The age in days at reference start date is calculated for each animal based on the age related variables in DM:

1. If BRTHDTC is populated compute DM.RFSTDTC – DM.BRTHDTC + 1
2. Else If AGE is populated convert from units specified in AGEU to days.
3. Else If AGETXT is populated convert the mid-point of the range from units specified in AGEU to days.
   These AGEU units are handled with the described conversion from value to number of days:
   - DAYS
   - WEEKS : value * 7
   - MONTHS : value * 365/12
   - YEARS : value * 365

If input parameter `inclUncertain=TRUE`, uncertain animals are included in the output set. These uncertain situations are identified and reported (in column UNCERTAIN_MSG):

- TX parameter 'TCNTRL' is missing
- TXVAL for TX parameter 'TCNTRL' cannot be identified as Negative or Positive control according to the algorithm described above

Value

The function return a data.table with columns:

- STUDYID (character)
- Additional columns contained in the `studyList` table
- TCNTRL (character)
  The value of the TX parameter TCNTRL which is used for identification of whether it is a negative control group or not
- USUBJID (character)
- RFSTDTC (character)
- DM_AGEDAYS (integer)
  The calculated age in days of the animal at the reference start day - i.e. the age registered in DM.
- DSDECOD (character)
  The standardized disposition term for the animal
- DS_AGEDAYS (integer)
  The calculated age in days of the animal at the disposition
- NO_AGE_MSG (character)
  Empty or contains the reason if a DM_AGEDAYS couldn't be calculated
- UNCERTAIN_MSG (character)
  Included when parameter `inclUncertain=TRUE`.
  Contains the reason for an uncertain animal is NA for rows for confident identified negative control animals.
- NOT_VALID_MSG (character)
  Included if the column is included in data table specified in `studyList`,

Examples

```
## Not run:
controlAnimals <- getControlSubj(myDbToken, allSTudies)

## End(Not run)
```

`getFindingsPhase` *Extract a set of findings for a specified study phase - or just add phase for each animal.*

Description

Returns a data table with the set of findings rows included in the `findings` of the phase(s) specified in the `phaseFilter`.
If the `phaseFilter` is empty (null, na or empty string), all rows from `findings` are returned with the an additional PHASE column.

Usage

```
getFindingsPhase(
  dbToken,
  findings,
  phaseFilter = NULL,
  inclUncertain = FALSE,
  noFilterReportUncertain = TRUE
)
```

Arguments

`dbToken`	Mandatory Token for the open database connection (see `initEnvironment`).
`findings`	Mandatory, data.table. A data.table with the set of finding rows to process. The table must include at least columns named • STUDYID • USUBJID • DOMAIN • domainSEQ • domainDTC where domain is the name of the actual findings domain - e.g. LBSEQ and LBDTC
`phaseFilter`	Optional, character. The phase value criterion to be used for filtering of the list of animals. It can be a single string, a vector or a list of multiple strings.
`inclUncertain`	Mandatory, boolean. Only relevant if the `phaseFilter` is not empty. Indicates whether finding rows for which the phase cannot be confidently identified shall be included or not in the output data table.

`noFilterReportUncertain`
: Mandatory, boolean.
Only relevant if the `phaseFilter` is empty.
Indicates if the reason should be included if the phase cannot be confidently decided for an animal.

Details

The logic for the extraction is based on the subject elements and the trial design domains - for each finding row:

- The related subject element is found in SE as the row where the value of domainDTC is within the interval from SESTDTC to SEENDTC
- The actual EPOCH is found in TA in the row matching the found element (via the ETCD value)
- The actual study phase is derived from the EPOCH value matching at set of text patterns

For pooled findings rows - i.e. POOLID is populated instead of USUBJID - the phase is identified per animal included in the each pool and finding, and if all identified phases are equal per pool and finding, the identified phase are returned per pool and finding.

The populated value of a phase is one of:

- 'Screening'
  If TA.EPOCH fulfills one:
  - contains 'pre' followed by one of ['treat','trt','dos',test','study','exposure']
  - contains one of ['acclimat','screen','baseline','allocat','random']
- 'Recovery'
  If TA.EPOCH doesn't fulfill the pattern for 'Screening' and fulfills one of:
  - contains 'recovery'
  - contains 'post' followed by one of ['treat','trt','dos','test','study','exposure']
- 'Treatment'
  If TA.EPOCH doesn't fulfill the patterns for 'Screening' or 'Recovery' and fulfills both:
  - contains one of ['treat','trt','dos','test','exposure']
  - does not contain any of ['off','non'|','free'|','holiday']
- 'Uncertain'
  If the TA.EPOCH is empty or does not fulfills any of the requirements described for the three phases above.

If input parameter `inclUncertain=TRUE`, findings rows where the phase cannot be confidently identified are included in the output set. These uncertain situations are identified and reported (in column UNCERTAIN_MSG):

- One of the date/time values SESTDTC, SEENDTC or domainDTC is empty or contains an invalid ISO 8601 value
- The value of domainDTC is included in more then one SESTDTC/SEENDTC interval
- The EPOCH value does not match any of the patterns identifying the set of possible study phases.

- Different phases have been identified for individual subjects in a pool for a given finding

The same checks are performed and reported in column NOT_VALID_MSG if `phaseFilter` is empty and `noFilterReportUncertain=TRUE`.

Value

The function returns a data.table with columns in this order:

- All columns contained in the `findings` input table (original order except optional UNCERTAIN_MSG and NOT_VALID_MSG)
- PHASE (character)
- UNCERTAIN_MSG (character)
  Included when parameter `inclUncertain=TRUE`.
  In case the phase cannot be confidently matched during the filtering of data, the column contains an indication of the reason.
  If any uncertainties have been identified for individual subjects included in pools for pooled finding rows, all messages for subjects per pool/findings are merged together and reported as one message per pool/finding.
  Is NA for rows where phase can be confidently matched.
  A non-empty UNCERTAIN_MSG value generated by this function is merged with non-empty UNCERTAIN_MSG values which may exist in the input set of findings specified in `findings` - separated by 'I'.
- NOT_VALID_MSG (character)
  Included when parameter `noFilterReportUncertain=TRUE`.
  In case the phase cannot be confidently decided, the column contains an indication of the reason.
  Is NA for rows where phase can be confidently decided.
  A non-empty NOT_VALID_MSG value generated by this function is merged with non-empty NOT_VALID_MSG values which may exist in the input set of findings `findings` - separated by 'I'.

Examples

```
## Not run:
# Extract LB rows for the Treatment phase - include uncertain rows
getFindingsPhase(dbToken, lb,
                 phaseFilter = 'Treatment',
                 inclUncertain = TRUE)
# No filtering, just add PHASE to FW rows - do not include messages when
# the phase cannot be confidently identified
getFindingsPhase(dbToken, fw,
                 noFilterReportUncertain = FALSE)

## End(Not run)
```

getFindingsSubjAge *Add the subject age at finding time - and optionally extract the set of findings within a specified range of age.*

Description

Returns a data table with the set of findings rows included in the findings where the age of subjects at finding time is within the interval specified in fromAge to fromAge.
If the fromAge and fromAge are empty (null, na or empty string), all rows from findings are returned.

Usage

```
getFindingsSubjAge(
  dbToken,
  findings,
  animalList,
  fromAge = NULL,
  toAge = NULL,
  inclUncertain = FALSE,
  noFilterReportUncertain = TRUE
)
```

Arguments

dbToken	Mandatory Token for the open database connection (see initEnvironment).
findings	Mandatory, data.table. A table with the set of input finding rows to process. The table must include at least columns named • STUDYID • USUBJID • DOMAIN • [domain]SEQ • [domain]DY • [domain]DTC where [domain] is the name of the actual findings domain - e.g. LBSEQ, LBDY and LBDTC
animalList	Mandatory, data.table. A data with the set of animals included in the findings table (may contain more animals than included in findings). The data set must contain at least these columns returned by the function getControlSubj • STUDYID • USUBJID

- RFSTDTC
- DM_AGEDAYS
- NO_AGE_MSG

`fromAge` Optional, character
The start of age interval to extract.
Must be in a string in this format:
`[value][age unit]` where `[age unit]` is one of

- `d, day, days`
- `w, week, weeks`
- `m, month, months`
- `y, year, years`

The unit is case-insensitive, space(s) between age value and unit is allowed.

`toAge` Optional. character
The start of age interval to extract.
Must be in a string in in the same format as described for `fromAge`.

`inclUncertain` Mandatory, boolean.
Only relevant if the `fromAge` and/or `toAge` is/are not empty.
Indicates whether finding rows for which the age at finding time cannot be confidently identified, shall be included or not in the output data table.

`noFilterReportUncertain`
Optional, boolean.
Only relevant if the `fromAge` and `toAge` are empty.
Indicates if the reason should be included if the age at finding time cannot be confidently decided for an animal.

Details

In both situation, the subject age at finding time is calculated into an additional column `AGEDAYS` for each row in `findings` combined with the the additional input data.table `animalList` using this algorithm:

- Determine the number of study days between study start and findings
  - if `findings.[domain]DY` is populated
    - If `findings.[domain]DY > 0` then use `findings.[domain]DY - 1`
    - Else use `findings.[domain]DY`
  - Else If `findings.[domain]DTC` is populated compute `animalList.RFSTDTC – findings.[domain]DTC` in days
    where `animalList.RFSTDTC` is each subject's reference start date (`DM.RFSTDTC`)
- Animal age at time of finding is then calculated as `animalList.AGEDAYS + [study days between study start and findings]`
  where `animalList.AGEDAYS` is the subject age at reference start date(calculated during extraction of control subjects in `getControlSubj`.
- For pooled findings rows - i.e. POOLID is populated instead of USUBJID - the animal age at time of finding is calculated per animal included in the each pool and finding.

  - If all calculated ages are equal within a pool and finding, the calculated age is populated for this pool/finding.
  - If all calculated ages are within the same time internal (2 days) within a pool and finding, the minimum calculated age plus 1 day is populated for this pool/finding.

If both fromAge and toAge values are specified - all the rows from the input table findings where value of the calculated AGEDYAS is within the interval of the specified start/end age interval are returned - including the values equal to the start/end age values.
If only a fromAge value is specified - all the rows from the input table findings where value of AGEDYAS equal to or greater than the input age are returned.
If only a toAge value is specified - all the rows from input table findings where value of AGEDAYS is equal to or less than the input age are extracted and returned. The input age value(s) is/are converted to days before extraction of rows from the input data tables using the input value(s) as filter - using this conversion:

- DAYS
- WEEKS : value * 7
- MONTHS : value * 365/12
- YEARS : value * 365

If input parameter inclUncertain=TRUE, findings rows where the age at finding time cannot be confidently identified are included in the output set. These uncertain situations are identified and reported (in column UNCERTAIN_MSG):

- No age at reference time has been calculated for subject (animalList.AGEDAYS)
- Reference start time is missing or contains invalid ISO8601 date value for subject (animalList.RFSTDTC).
- Missing [domain]DY value and missing or invalid ISO8601 date [domain]DTC value for finding
- For pooled findings:
  - More than two days between minimum and maximum of animalList.AGEDAYS for the set of animals in a pool.
  - Different values in animalList.RFSTDTC for the set of animals in a pool.

The same checks are performed and reported in column NOT_VALID_MSG if fromAge and fromAge are empty and noFilterReportUncertain = TRUE.

Value

The function returns a data.table with columns in this order:

- All columns contained in the findings input table (original order except optional UNCERTAIN_MSG and NOT_VALID_MSG)
- AGEDAYS (character)
  The subject age at finding time calculated in days. Is NA if thge age cannot be confidently calculated.

- UNCERTAIN_MSG (character)
  Included when parameter inclUncertain=TRUE.
  In case the age at finding time cannot be confidently matched during the filtering of data, the column contains an indication of the reason.
  If any uncertainties have been identified for individual subjects included in pools for pooled finding rows, one message for is reported per pool/finding.
  Is NA for rows where the age at finding time can be confidently matched.
  A non-empty UNCERTAIN_MSG value generated by this function is merged with non-empty UNCERTAIN_MSG values which may exist in the input set of findings specified in findings - separated by '|'.
- NOT_VALID_MSG (character)
  Included when parameter noFilterReportUncertain=TRUE.
  In case the age at finding time cannot be confidently calculated, the column contains an indication of the reason.
  Is NA for rows where age at finding time can be confidently calculated.
  A non-empty NOT_VALID_MSG value generated by this function is merged with non-empty NOT_VALID_MSG values which may exist in the input set of findings findings - separated by '|'.

Examples

```
## Not run:
# Extract LB rows for the animals at age between 8 and 12 weeks at finding
# time - include uncertain rows
getFindingsSubjAge(dbToken = db,
                   findings = lb,
                   animalList = animals,
                   fromAge = '8w',
                   toAge = '12w',
                   inclUncertain = TRUE)
# No filtering, just add AGEDAYS to FW rows - do not include messages when
# the AGEDAYS cannot be confidently identified
getFindingsSubjAge(dbToken = db, findings = fw,  animalList = animals,
                   noFilterReportUncertain = FALSE)

## End(Not run)
```

---

getStudiesSDESIGN *Extract a list of SEND studies with a specified study design - or just add actual study design for each study.*

Description

Returns a data table with the list of study ids from TS where the value of TSVAL for the TSPARMCD 'SDESIGN' is equal to a given study design.
If the studyDesignFilter is empty (null, na or empty string) - all rows for the TSPARMCD 'SDESIGN' are returned.

Usage

```
getStudiesSDESIGN(
  dbToken,
  studyList = NULL,
  studyDesignFilter = NULL,
  exclusively = TRUE,
  inclUncertain = FALSE,
  noFilterReportUncertain = TRUE
)
```

Arguments

`dbToken`	Mandatory. Token for the open database connection (see `initEnvironment`).
`studyList`	Optional, data.table. A table with the list of studies to process. If empty, all studies in the data base are processed The table must include at least a column named 'STUDYID'
`studyDesignFilter`	Mandatory, character. The study design to use as criterion for filtering of the study id values. It can be a single string, a vector or a list of multiple strings.
`exclusively`	Mandatory, boolean. • TRUE: Include studies only for studies with no other study design(s) than included in `studyDesignFilter`. • FALSE: Include all studies with study design matching `studyDesignFilter`.
`inclUncertain`	Mandatory, boolean. Indicates whether study ids with SDESIGN value which are is missing or wrong shall be included or not in the output data table.
`noFilterReportUncertain`	Mandatory, boolean Only relevant if the `studyDesignFilter` is empty. Indicates if the reason should be included if the SDESIGN cannot be confidently decided for an animal.

Details

Extracts the set of studies from TS where the value of TSVAL for the TSPARMCD 'SDESIGN' is equal to a given study design.
The comparison of study design values are done case insensitive.

If a data table with a list of studies is specified in `studyList`, only the subset of studies included in that set is processed.

If input parameter `inclUncertain=TRUE`, uncertain animals are included in the output set. These uncertain situations are identified and reported (in column UNCERTAIN_MSG):

- without any row for TSPARMCD='SDESIGN' or

- TSVAL doesn't contain a value included in the CDISC CT list 'DESIGN' for TSPARMCD='SDESIGN' (case insensitive comparison)

The same checks are performed and reported in column NOT_VALID_MSG if `studyDesignFilter` is empty and `noFilterReportUncertain=TRUE`.

Value

The function returns a data.table with columns:

- STUDYID (character)
- Additional columns contained in the `studyList` table (if such an input table is given)
- SDESIGN (character)
  If multiple TSPARMCD 'SDESIGN' values are extracted for a studies, all the values are merged into a comma separated string.
- UNCERTAIN_MSG (character)
  Included when parameter `inclUncertain=TRUE`.
  Contains indication of whether STSTDTC is missing or has wrong format.
  Is NA for rows where SDESIGN is valid.
  A non-empty UNCERTAIN_MSG value generated by this function is merged with non-empty UNCERTAIN_MSG values which may exist in the optional input set of studies specified in `studyList` - separated by '|'.
- NOT_VALID_MSG (character)
  Included when parameter `noFilterReportUncertain=TRUE`.
  In case the SDESIGN cannot be confidently decided, the column contains an indication of the reason.
  Is NA for rows where SDESIGN can be confidently decided.
  A non-empty NOT_VALID_MSG value generated by this function is merged with non-empty NOT_VALID_MSG values which may exist in the input set of studies specified in `studyList` - separated by '|'.

Examples

```
## Not run:
GetStudyListSDESIGN(myDbToken, 'PARALLEL')

## End(Not run)
```

`getStudiesSTSTDTC` *Extract a list of SEND studies with study start date within a specified interval - or just add actual study start date for each study*

Description

Returns a data table with the list of study ids from TS where the value of TSVAL for the TSPARMCD 'STSTDTC' is within a a given date interval.
If the `fromDTC` and`toDTC` are empty (null, na or empty string)

- all rows for the TSPARMCD 'STSTDTC' are returned.

Usage

```
getStudiesSTSTDTC(
  dbToken,
  studyList = NULL,
  fromDTC = NULL,
  toDTC = NULL,
  inclUncertain = FALSE,
  noFilterReportUncertain = TRUE
)
```

Arguments

`dbToken`	Mandatory. Token for the open database connection (see `initEnvironment`).
`studyList`	Optional. A data.table with the list of studies to process. If empty, all studies in the data base are processed The table must include at least a column named 'STUDYID'.
`fromDTC`	Optional (either or both of `fromDTC` and `toDTC` must be filled). The start of the date interval to extract - must be in ISO8601 date format.
`toDTC`	Optional (either or both of `fromDTC` and `toDTC` must be filled). The end of the date interval to extract - must be in ISO8601 date format.
`inclUncertain`	Mandatory, boolean. Indicates whether study ids with STSTDTC which are are missing or wrong shall be included or not in the output data table.
`noFilterReportUncertain`	Mandatory, boolean Only relevant if the `fromDTC` and`toDTC` are empty. Indicates if the reason should be included if the STSTDTC cannot be confidently decided for an animal.

Details

Extracts the set of study ids from TS where the value of TSVAL for the TSPARMCD 'STSTDTC' falls within a specified start/end date interval in IS8601 format (input parameters `fromDTC`/`toDTC`).

Both complete and incomplete input start/end dates can be handled.

- If only a year is specified - the date set to the first of January that year.
- If only a year and month is specified - the date set to the first day in that month.
- If a time part is included in a specified input start/end date, it is ignored.

If both a start and end input date are specified - all the STUDYID values from TS where TSVAL for TSPARMCD 'STSTDTC' is with the interval of the specified start/end date interval are extracted and returned - including the values equal to the start/end dates. are included.

If only a start input date is specified - all the STUDYID values from TS where TSVAL for TSPARMCD 'STSTDTC' is equal to or later than the input date are extracted and returned.

If only an end date is specified - all the STUDYID values from TS where TSVAL for TSPARMCD 'STSTDTC' is equal to or earlier than the are date are extracted and returned.

If a data table with a list of studies is specified in `studyList`, only the subset of studies included in that set is processed.

If input `inclUncertain` is TRUE, uncertain studies are included in the output set. These uncertain situations are identified and reported (in column UNCERTAIN_MSG):

- TS contains now row for TSPARMCD='STSTDTC'
- TSVAL contains an invalid ISO8601 date format for TSPARMCD='STSTDTC'

The same checks are performed and reported in column NOT_VALID_MSG if `fromDTC` and `toDTC` are empty and `noFilterReportUncertain=TRUE`.

Value

The function return a data.table with columns:

- STUDYID (character)
- Additional columns contained in the `studyList` table (if such an input table is given)
- STSTDTC (character - ISO8601 format)
- UNCERTAIN_MSG (character)
  Only included when parameter `inclUncertain=TRUE`.
  Contains indication of whether STSTDTC is missing of has wrong format.
  Is NA for rows where SDESIGN is valid.
  A non-empty UNCERTAIN_MSG value generated by this function is merged with non-empty UNCERTAIN_MSG values which may exist in the optional input set of studies specified in `studyList` - separated by '|'.
- NOT_VALID_MSG (character)
  Included when parameter `noFilterReportUncertain=TRUE`.
  In case the STSTDTC cannot be confidently decided, the column contains an indication of the reason.
  Is NA for rows where STSTDTC can be confidently decided.
  A non-empty NOT_VALID_MSG value generated by this function is merged with non-empty NOT_VALID_MSG values which may exist in the input set of studies specified in `studyList` - separated by '|'.

Examples

```
## Not run:
GetStudyListSTSTDTC(myDbToken, allSTudies, '2018','2020')

## End(Not run)
```

getSubjData *Extract data from a subject level domain.*

Description

Extracts and returns all rows from the specified domain for the set of subjects included in animalList.

Usage

```
getSubjData(dbToken, animalList, domain, colList = NULL)
```

Arguments

dbToken	Mandatory Token for the open database connection (see initEnvironment).
animalList	Mandatory, data.table. A table with the list of animals to be included in the output data. The table must include at least columns named 'STUDYID' and 'USUBJID'.
domain	Mandatory, character, not case sensitive. The name of the domain table to extract data from. The name must be a subject level domain - i.e. a table including a 'USUBJID' column.
colList	Optional, character, not case sensitive. The list of columns to be extracted from the specified domain table. It can be a single string, a vector or a list of multiple strings.

Value

The function returns a data.table with all the rows for the animals included in animalList.
If no columns have been specified in colList, all the columns in the table colList are included.
If a list of columns have been specified in colList, these are included. In addition, a set of columns are always included, whether they are included in colList or not:

- To ensure each row can be uniquely identified:
  - DOMAIN
  - STUDYID
  - USUBJID
  - POOLID (if it exists)
  - domainSEQ (if it exists)
- For finding tables - to support age calculation and evaluation of study phase:
  - domainDTC
  - domainDY

The order of the columns are as they are defined for the domain in the SEND IG.
The data table contains both

- subject level data - i.e. rows where USUBJID is not empty
- if applicable for the domain, pool level data - i.e. rows where POOLID is not empty.
  In this case, all pools, which includes any of the subjects included in animalList, are included

Examples

```
## Not run:
# Extract all columns from DM:
getSubjData(myDbToken, myControlAnimals, 'dm')

# Extract selected columns from LB:
getSubjData(myDbToken, myControlAnimals, 'LB',
            list('LBTESTCD', 'LBCAT',
                 'LBSTRESC', 'LBSTRESN', 'LBSTRESU',
                 'LBSTAT', 'LBREASND',
                 'LBTPT'))

## End(Not run)
```

---

getSubjRoute *Extract the set of animals of the specified route of administration - or just add actual route of administration for each animal.*

Description

Returns a data table with the set of animals included in the animalList matching the route of administration specified in the routeFilter.
If the routeFilter is empty (null, na or empty string) - all rows from animalList are returned with an additional populated ROUTE column.

Usage

```
getSubjRoute(
  dbToken,
  animalList,
  routeFilter = NULL,
  exclusively = FALSE,
  matchAll = FALSE,
  inclUncertain = FALSE,
  noFilterReportUncertain = TRUE
)
```

Arguments

dbToken	Mandatory Token for the open database connection (see initEnvironment).

`animalList` Mandatory, data.table.
A table with the list of animals to process.
The table must include at least columns named 'STUDYID' and 'USUBJID'.

`routeFilter` Optional, character.
The route of administration value(s) to use as criterion for filtering of the input data table.
It can be a single string, a vector or a list of multiple strings.

`exclusively` Mandatory if `routeFilter` is non empty, boolean.

- TRUE: Include animals only for studies with no other routes then included in `routeFilter`.
- FALSE: Include animals for all studies with route matching `routeFilter`.

`matchAll` Mandatory if `routeFilter` is non empty, boolean.

- TRUE: Include animals only for studies with route(s) matching all values in `routeFilter`.
- FALSE: Include animals for all studies with route matching at least one value in `routeFilter`.

`inclUncertain` Mandatory if `routeFilter` is non empty, boolean,.
Indicates whether animals for which the route cannot be confidently identified shall be included or not in the output data table.

`noFilterReportUncertain`
Mandatory if `routeFilter` is empty, boolean
Only relevant if the `routeFilter` is empty.
Indicates if the reason should be included if the route cannot be confidently decided for an animal.

Details

The route of administration per animal are identified by a hierarchical lookup in these domains

- EX - If a distinct not empty EXROUTE value is found for animal, this is included in the output.
- TS - if a distinct TS parameter 'ROUTE' value exists for the study, this is included in the output.

The comparison of route values is done case insensitive and trimmed for leading/trailing blanks.

If input parameter `inclUncertain=TRUE`, uncertain animals are included in the output set. These uncertain situations are identified and reported (in column UNCERTAIN_MSG):

- TS parameter ROUTE is missing for study and no EX rows contain a EXROUTE value for the animal
- The selected EXROUTE or TS parameter ROUTE value is invalid (not CT value - CDISC SEND code list ROUTE)
- Multiple EXROUTE values have been found for the animal
- Multiple TS parameter ROUTE values are registered for study but no EX rows contain a EXROUTE value for the animal

- The found EXROUTE value for animal is not included in the TS parameter ROUTE value(s) registered for study

The same checks are performed and reported in column NOT_VALID_MSG if `routeFilter` is empty and `noFilterReportUncertain=TRUE`.

Value

The function returns a data.table with columns:

- STUDYID (character)
- Additional columns contained in the `animalList` table
- ROUTE (character)
  The value is always returned in uppercase and trimmed for leading/trailing blanks.
- UNCERTAIN_MSG (character)
  Included when parameter `inclUncertain=TRUE`.
  In case the ROUTE cannot be confidently matched during the filtering of data, the column contains an indication of the reason.
  Is NA for rows where ROUTE can be confidently matched.
  A non-empty UNCERTAIN_MSG value generated by this function is merged with non-empty UNCERTAIN_MSG values which may exist in the input set of animals specified in `animalList` - separated by 'I'.
- NOT_VALID_MSG (character)
  Included when parameter `noFilterReportUncertain=TRUE`.
  In case the ROUTE cannot be confidently decided, the column contains an indication of the reason.
  Is NA for rows where the ROUTE can be confidently decided.
  A non-empty NOT_VALID_MSG value generated by this function is merged with non-empty NOT_VALID_MSG values which may exist in the input set of animals `animalList` - separated by 'I'.

Examples

```
## Not run:
# Extract animals administered oral or oral gavage plus uncertain animals
getSubjRoute(dbToken, controlAnimals,
             routeFilter = c('ORAL', 'ORAL GAVAGE'),
             inclUncertain = TRUE)
# Extract animals administered oral or oral gavage.
# Do only include studies which include both route values
getSubjRoute(dbToken, controlAnimals,
             routeFilter = c('ORAL', 'ORAL GAVAGE'),
             matchAll = TRUE)
# Extract animals administered subcutaneous.
# Include only animals from studies which do not contain other route values
getSubjRoute(dbToken, controlAnimals,
             routeFilter = 'subcutaneous',
             exclusively = TRUE)
# No filtering, just add ROUTE - do not include messages when
# these values cannot be confidently found
```

```
getSubjRoute(dbToken, controlAnimals,
             noFilterReportUncertain = FALSE)

## End(Not run)
```

`getSubjSex` *Extract the set of animals of the specified sex - or just add the sex of each animal.*

Description

Returns a data table with the set of animals included in the `animalList` of the sex specified in the `sexFilter`.
If the `sexFilter` is empty (null, na or empty string) - all rows from `animalList` are returned with the an additional populated SEX column.

Usage

```
getSubjSex(
  dbToken,
  animalList,
  sexFilter = NULL,
  inclUncertain = FALSE,
  noFilterReportUncertain = TRUE
)
```

Arguments

`dbToken`	Mandatory Token for the open database connection (see `initEnvironment`).
`animalList`	Mandatory, data.table. A table with the list of animals to process. The table must include at least columns named 'STUDYID' and 'USUBJID'.
`sexFilter`	Optional, character. The sex value criterion to be used for filtering of the list of animals. It can be a single string, a vector or a list of multiple strings.
`inclUncertain`	Mandatory, boolean. Indicates whether animals for which the sex cannot be confidently identified shall be included or not in the output data table.
`noFilterReportUncertain`	Mandatory, boolean. Only relevant if the `sexFilter` is empty. Indicates if the reason should be included if the sex cannot be confidently decided for an animal.

Details

The sex value is decided from the DM.SEX variable.
The comparison of DM.SEX with the given value(s) in `sexFilter` is done case-insensitive.

If input parameter `inclUncertain=TRUE`, uncertain animals are included in the output set. These uncertain situations are identified and reported (in column UNCERTAIN_MSG):

- The DM.SEX value is empty or invalid (not CT value - CDISC codelist SEX - case insensitive comparison)

The same checks are performed and reported in column NOT_VALID_MSG if `sexFilter` is empty and `noFilterReportUncertain=TRUE`.

Value

The function returns a data.table with columns:

- STUDYID (character)
- Additional columns contained in the `animalList` table
- SEX (character)
- UNCERTAIN_MSG (character)
  Included when parameter `inclUncertain=TRUE`.
  In case the sex cannot be confidently matched during the filtering of data, the column contains an indication of the reason.
  Is NA for rows where SEX can be confidently matched.
  A non-empty UNCERTAIN_MSG value generated by this function is merged with non-empty UNCERTAIN_MSG values which may exist in the input set of animals specified in `animalList` - separated by 'I'.
- NOT_VALID_MSG (character)
  Included when parameter `noFilterReportUncertain=TRUE`.
  In case the sex cannot be confidently decided, the column contains an indication of the reason.
  Is NA for rows where sex can be confidently decided.
  A non-empty NOT_VALID_MSG value generated by this function is merged with non-empty NOT_VALID_MSG values which may exist in the input set of animals `animalList` - separated by 'I'.

Examples

```
## Not run:
getSubjSex(myDbToken, controlAnimals, 'M')

## End(Not run)
```

`getSubjSpeciesStrain` *Extract the set of animals of the specified species and strain - or just add the species and strain for each animal.*

Description

Returns a data table with the set of animals included in the `animalList` matching the species and strain specified in the `speciesFilter` and `strainFilter`.
If the `speciesFilter` and `strainFilter` are empty (null, na or empty string) - all rows from `animalList` are returned with additional populated SPECIES and STRAIN columns.

Usage

```
getSubjSpeciesStrain(
  dbToken,
  animalList,
  speciesFilter = NULL,
  strainFilter = NULL,
  inclUncertain = FALSE,
  exclusively = FALSE,
  noFilterReportUncertain = TRUE
)
```

Arguments

`dbToken`	Mandatory Token for the open database connection (see `initEnvironment`).
`animalList`	Mandatory, data.table. A table with the list of animals to process. The table must include at least columns named 'STUDYID' and 'USUBJID'.
`speciesFilter`	Optional, character. The species value(s) to use as criterion for filtering of the input data table. It can be a single string, a vector or a list of multiple strings.
`strainFilter`	Optional, character. The strain value(s) to use as criterion for filtering of the input data table. It is only valid to specify value(s) if one or more values have been specified for parameter `speciesFilter` It can be a single string, a vector or a list of multiple strings. When multiple values are specified for `speciesFilter`, each strain value must be prefixed by species and ':' , e.g. `c('RAT:WISTAR','DOG: BEAGLE')`. There may be included any number of blanks after ':'
`inclUncertain`	Mandatory, boolean. Indicates whether animals for which the species or strain cannot be confidently identified shall be included or not in the output data table.
`exclusively`	Mandatory, boolean.

- TRUE: Include animals only for studies with no other species and optional strains then included in `speciesFilter` and `strainFilter`
- FALSE: Include animals for all studies with species and strain matching `speciesFilter` and `strainFilter` respectively.

`noFilterReportUncertain`
: Optional, boolean.
Only relevant if the `speciesFilter` and `strainFilter` are empty.
Indicates if the reason should be included if the species or strain cannot be confidently decided for an animal.

Details

The species and strain per animal respectively are identified by a hierarchical lookup in these domains

- DM - If the DM.SPECIES (DM.STRAIN) isn't empty, this value is included in the output.
- TX - if a TX parameter 'SPECIES' ('STRAIN') exists for the group related to the animal, the TXVAL value for this is included in the output.
- TS - if a TS parameter 'SPECIES' ('STRAIN') exists, this is included in the output.

The comparisons of species/strain values is done case insensitive and trimmed for leading/trailing blanks.

If input parameter `inclUncertain=TRUE`, uncertain animals are included in the output set. These uncertain situations are identified and reported for SPECIES and STRAIN respectively (in column UNCERTAIN_MSG):

- TS parameter SPECIES/STRAIN is missing or invalid (not CT value - CDISC SEND code list SPECIES/STRAIN) and TX parameter SPECIES/STRAIN is missing or invalid (not CT value) and DM.SPECIES/STRAIN is missing or invalid (not CT value)
- Different values of SPECIES/STRAIN across TS, TX and DM for studies where no or only one TS parameter SPECIES/STRAIN is registered
- Multiple TS parameter SPECIES/STRAIN values are registered for study and TX parameter SPECIES/STRAIN and/or DM.SPECIES/STRAIN do not match any of the TS values.
- Multiple TS parameter SPECIES/STRAIN values are registered for study and TX parameter SPECIES/STRAIN and DM.SPECIES/STRAIN are unequal.

The same checks are performed and reported in column NOT_VALID_MSG if `speciesFilter` and `strainFilter` are empty and `noFilterReportUncertain=TRUE`.

Value

The function returns a data.table with columns:

- STUDYID (character)
- Additional columns contained in the `animalList` table

- SPECIES (character) The value is always returned in uppercase and trimmed for leading/trailing blanks.
- STRAIN (character) The value is always returned in uppercase and trimmed for leading/trailing blanks.
- UNCERTAIN_MSG (character)
  Included when parameter `inclUncertain=TRUE`.
  In case the species or strain cannot be confidently matched during the filtering of data, the column contains an indication of the reason.
  Is NA for rows where species and strain can be confidently matched.
  A non-empty UNCERTAIN_MSG value generated by this function is merged with non-empty UNCERTAIN_MSG values which may exist in the input set of animals specified in `animalList` - separated by 'I'.
- NOT_VALID_MSG (character)
  Included when parameter `noFilterReportUncertain=TRUE`.
  In case the species or strain cannot be confidently decided, the column contains an indication of the reason.
  Is NA for rows where species and strain can be confidently decided.
  A non-empty NOT_VALID_MSG value generated by this function is merged with non-empty NOT_VALID_MSG values which may exist in the input set of animals `animalList` - separated by 'I'.

Examples

```
## Not run:
# Extract rats and mice plus uncertain animals
getSubjSpeciesStrain(dbToken, controlAnimals,
                     speciesFilter = c('RAT', 'MOUSE'),
                     inclUncertain = TRUE)
# Extract Spargue-Dawley rats plus uncertain animals.
# Include only animals from studies which do not contain other species or
# strains
getSubjSpeciesStrain(dbToken, controlAnimals,
                     speciesFilter = 'RAT',
                     strainFilter = 'SPRAGUE-DAWLEY',
                     inclUncertain = TRUE,
                     exclusively = TRUE,
                     noFilterReportUncertain = TRUE)
# Extract Wistar rats and and Beagle dogs - and no uncertain animals
getSubjSpeciesStrain(dbToken, controlAnimals,
                     speciesFilter = c('RAT', 'DOG'),
                     strainFilter = c('RAT: WISTAR', 'DOG: BEAGLE'))
# No filtering, just add SPECIES and STRAIN - do not include messages when
# these values cannot be confidently found
getSubjSpeciesStrain(dbToken, controlAnimals,
                     noFilterReportUncertain = FALSE)

## End(Not run)
```

getTabColLabels *Get labels for columns in a data.table*

Description

Get labels for columns in a data.table

Usage

```
getTabColLabels(table)
```

Arguments

table	Mandatory The data.table to get column labels for

Value

A named vector with each column/label pair. If a column have no defined label, the label is 'na'

Examples

```
## Not run:
colLabels = getTabColLabels(controlAnimalsAll)

## End(Not run)
```

initEnvironment *Initialize the environment.*

Description

Open or create a SEND database and return a token for the open database connection.

Usage

```
initEnvironment(
  dbType = NULL,
  dbPath = NULL,
  dbHost = NULL,
  dbCreate = FALSE,
  dbUser = NULL,
  dbPwd = NULL,
  dbPort = NULL,
  dbSchema = NULL,
  ctFile = NULL
)
```

Arguments

dbType	Mandatory, character The type of database, valid values (case insensitive): • 'sqlite' • 'oracle' • 'postgresql'
dbPath	Mandatory, character The path to the database (path to file or another kind of db reference)
dbHost	Optional, character Name of PostgreSQL host/server. This parameter is only relevant to specify if a server or host name is necessary to connect to the database. Only necessary for a PostgreSQL database connection. Add param for port number
dbCreate	Mandatory, boolean If TRUE, a new database is to be created - this is only valid for dbType 'sqlite'
dbUser	Mandatory, character - if login credentials are required for the specific db type The user name to be used for login to database.
dbPwd	Mandatory, character - if login credentials are required for the specific db type The password to be used for login to database.
dbPort	Optional, character? Port number for connecting to database. This parameter is only necessary if the database type requires a port number for connection (e.g., PostgreSQL uses port number 5432).
dbSchema	Optional, character The table owner of the SEND table in the specific database. This parameter is only relevant to specify if it is necessary to prefix table names with schema in SQL statements in the database.
ctFile	Optional, character. Name (full path) of CDISC CT file in Excel xlsx format to be imported. Only relevant to use if another CDISC CT version than the version included in packages is wanted. Add param for host/server

Details

If the function is executed with parameter dbCreate=FALSE (default), a connection to the specified database is opened. Dependent of the type of database (parameter dbType), a login using specified user credentials (parameters dbUser and dbPwd) may be done.
The database must contain a set of tables representing the SEND domains compliant with SEND IG version 3.0 and/on 3.1.

If the function is executed with parameter dbCreate=TRUE, an empty database is created and opened. This is only supported for a SQLite database, i.e., parameter dbType='sqlite'. The SEND domain tables may then be created by execution of the function dbCreateSchema.

Besides the open database connection, a set of CDISC SEND controlled terminology values are imported. If parameter `ctFile` is specified with a path to an Excel file containing a CDISC SEND ct version downloaded from https://evs.nci.nih.gov/ftp1/CDISC/SEND/, the content from this file is imported and used by some of the package's functions. Else a set of CDISC SEND CT values which are included in the packages is used by the package's functions. It's the newest CDISC SEND CT version at the time of the build of the current version of the package which is included.

Value

The function returns a token which is a data structure describing the open database connection. This token must be given as input parameter to all functions accessing the actual database.

Examples

```
## Not run:
db <- initEnvironment(dbType='sqlite',
                      dbPath='//servername/SendData/db/send.db',
                ctFile='//servername/SendData/metadata/SEND_Terminology_2019-12-27.xls')

db <- initEnvironment(dbType='oracle',
                      dbPath='dbserver:1521/send_db',
                      dbUser='ME',
                      dbPwd='mypassword',
                      dbSchema = 'send',
                ctFile='//servername/SendData/metadata/SEND_Terminology_2019-12-27.xls')

db <- initEnvironment(dbType='postgresql',
                      dbPath='send_db_name',
                      dbHost='dbserver',
                      dbUser='ME',
                      dbPwd='mypassword',
                      dbPort='5432',
                ctFile='//servername/SendData/metadata/SEND_Terminology_2019-12-27.xls')


## End(Not run)
```

---

standardize_file *Standardizes SEND xpt files using CDISC controlled terminologies*

Description

Standardizes SEND xpt files using CDISC controlled terminologies

Usage

```
standardize_file(input_xpt_dir, output_xpt_dir, json_file)
```

Arguments

`input_xpt_dir`	Mandatory. input folder name with xpt files under the folder.
`output_xpt_dir`	Mandatory. output folder name for writing the cleaned xpt files.
`json_file`	Mandatory. json filename used for mapping.

Value

No return value, called for side effects

Index

dbCreateIndexes, 2
dbCreateSchema, 3, *35*
dbDeleteStudies, 4
dbImportOneStudy, 5, *7*
dbImportStudies, 7
disconnectDB, 9

execSendDashboard, 9

gen_vocab, 10
genericQuery, 11
getControlSubj, 11, *17, 18*
getFindingsPhase, 14
getFindingsSubjAge, 17
getStudiesSDESIGN, 20
getStudiesSTSTDTC, 22
getSubjData, 25
getSubjRoute, 26
getSubjSex, 29
getSubjSpeciesStrain, 31
getTabColLabels, 34

initEnvironment, *2–5*, *7*, *9*, *11, 12*, *14*, *17*, *21*, *23*, *25, 26*, *29*, *31*, 34

standardize_file, 36