Package 'rCPRD'

November 18, 2020

Type Package

Title Drug Analysis for Electronic Healthcare Records

Version 0.1.13

Author Anthony Nash

Maintainer The package maintainer <anthony.nash@ndcn.ox.ac.uk>

Description (pre-release - test only) An R package for CPRD electronic healthcare records drug prescription analysis. Contains functions for the curation and manipulation of CPRD data, to save and load records, and for the analysis of records. CPRD records are expected to be loaded into the R environment, further information is available in the accompanying manuscript and on the Github page https://github.com/acnash/RDrugTrajectory

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports plyr,
dplyr,
foreach,
doParallel,
data.table,
parallel,
splus2R,
rlist,
reda,
ggplot2,
ggalluvial,
stats,
utils,
useful

Suggests knitr,
rmarkdown

VignetteBuilder knitr

Depends R (>= 2.10)

R topics documented:

addCovariateBooleanList *Not for external call. Adds time-independent TRUE/FALSE covariate values to an existing cox data.frame.*

Description

This function appends covariate columns to an existing cox data.frame (having called constructSurvivalTimeline first) for all those codes presented in the named covariateBooleanList argument. The covariate columns are named after the named list entries in the covariateBooleanList. The covariate search can be restricted to desired patients by providing a list of patient patids in the patidIDVector. Whether a patient's record has a desired code is determined by searching through the the medHistoryDF.

Usage

```
addCovariateBooleanList(
  coxDF,
  medHistoryDF,
  covariateBooleanList,
  patidVector = NULL
)
```

Arguments

coxDF	Dataframe generated by first calling constructSurvivalTimeline().
medHistoryDF	Dataframe of patient medical (clinical, therapy etc) data.
covariateBooleanList	A named list of codes (can be a mix of medcodes and prodcodes). Each named element is used as the column name for a covariate.
patidVector	a list or vector of patient patids. If NULL (default) then the medHistoryDF is filtered for all patients in the coxDF.

Details

The covariateBooleanList is a named list of medcodes and/or prodcodes e.g., List(drugA=c(1,2,3,4), drugB=c(5,6,7,8)).

Value

Dataframe similar to the coxDF argument but with additional columns to denote each boolean covariate. Values are 0 for not present, 1 for present.

ageGenderDF *Fabricated age and gender data.*

Description

Fabricated age and gender data stored in an R Dataframe.

Usage

```
data(testTherapyDF)
```

Format

A Dataframe object with columns: patid, yob, gender. As a strict requirement these columns must be included yet the user can add additional columns without concerns over package function compatibility. Yob (year of birth) is the age calculated taking into account the year the patient was born and the release year of the database.

Source

This is fabricated data. It takes the likeness of data from https://www.cprd.com/ after permission from ISAC.

References

NA

Examples

```
str(ageGenderDF)
```

calculateSurvivalTime *Not for external call: calculates survival time for a single patient.*

Description

Not for external call: calculates survival time for a single patient.

Usage

```
calculateSurvivalTime(indDF, indexVector, indexPosition, eventVector, endDate)
```

Arguments

indDF	Dataframe for a single patient.
indexVector	vector of codes that denote the index date for the patient.
indexPosition	A string denoting either "FIRST" or "LAST" for an index event before the time-to-event. Useful for "first drug X before event A" and "last drug X before event A".
eventVector	vector of codes that denote the event for the patient.
endDate	Date indicating when the study ended. Used to determine lost to follow-up patients.

Value

List with the named elements: survivalTime, indexDate and eventDate.

`calculateTDSurvivalTime` *Not for external call: calculates time-dependent survival time for a single patient.*

Description

If you wish to know more consult the constructSurvivalTimeline documentation.

Usage

```
calculateTDSurvivalTime(
  indDF,
  indexVector,
  indexPosition,
  eventVector,
  tdCovariateList,
  tdCovariateBehaviourVector,
  obsTime,
  endDate
)
```

Arguments

`indDF`	Dataframe for a single patient.
`indexVector`	vector of codes that denote the index date for the patient.
`indexPosition`	A string denoting either "FIRST" or "LAST" for an index event before the time-to-event. Useful for "first drug X before event A" and "last drug X before event A".
`eventVector`	vector of codes that indicate an event.
`tdCovariateList`	List of covariate vectors. Each covariate vector must be named. e.g., List(nameA=codeVectorA)
`tdCovariateBehaviourVector`	Whether to increment a covariate value or to stay fixed. The List structure must follow that of tdCovariateList.
`obsTime`	Number of days to observe from index date before stop recording event.
`endDate`	Date indicating when the study ended. Used to determine lost to follow-up patients.

Value

List with the named elements: survivalTime, indexDate and eventDate.

checkCPRDRecord *Checks whether a CPRD data frame is formatted well.*

Description

Every CPRD data frame must have at least the following columns: patid, eventdate, medcode/prodcode.

Usage

```
checkCPRDRecord(df, dataType = NULL)
```

Arguments

df	data frame to check.
dataType	A character vector of either "clinical", "referral" or "therapy." NULL as default, which throws a warning.

Value

A logical value. TRUE indicates a well formatted data frame, FALSE otherwise.

Examples

```
checkBool <- checkCPRDRecord(testTherapyDF, "therapy")
```

combinePrescriptionFrequencies *Combine multiple prescription frequencies under a common name.*

Description

Use to combine the drug prescription frequencies from a data frame of descriptions and codes and to give those combined frequencies a new name, for example, "All Amitriptyline"; a type of drug may have several drug codes, each with a unique description for example, a product name and dose.

Usage

```
combinePrescriptionFrequencies(df, structureList, rowIndices = FALSE)
```

Arguments

df	A data.frame with at least the columns "description" and "Frequency", such as drugDistributionDF from getFirstDrugPrescription.
structureList	A named list of numeric vectors. Each vector holds the drug prodcode or the row indices of the df row in which you want to sum Frequencies and return as a single entry.
rowIndices	A logical value. When TRUE it reads the structureList numeric vectors as data.frame row indices. If FALSE (default) it treats the numeric vectors as prodcodes.

Value

A data.frame with the same structure as the input df (description | code | Frequency), using the new description from the named structureList and the combined Frequencies. The code entry is NA. A NULL is returned if anything went wrong.

Examples

```
fileLocation <- NULL
firstDrugObj <- getFirstDrugPrescription(testTherapyDF,
idList=NULL,
prodCodesVector=NULL,
descriptionFile=fileLocation)

structureList <- list(
  Amitriptyline = c(139,209,31123,1395,2234,1425,1146,1147),
  Propranolol = c(145,5,41,8776,1166,3945,4493,543)
)
combinedDF <- combinePrescriptionFrequencies(firstDrugObj$drugDistributionDF,
                                             structureList)

#This totals the frequencies of those grouped product codes into the one
#entry and uses the list element name to create a new data frame entry.

#As row indices:
structureList <- list(
 Amitriptyline = c(1,2,3,13,22,45,46,47),
 Propranolol = c(4,5,6,7,11,41,42,43)
)
combinedDF <- combinePrescriptionFrequencies(firstDrugObj$drugDistributionDF,
                                             structureList,
                                             rowIndices=TRUE)

#This combines the entries at index 1,2,3,13,22,45,46,47 into the one entry
#and be named "Amitriptyline". If any indices are out of scope for the
#accompanying therapeutic df argument, they are ignored with a warning.
```

`constructMCF` *Build observation data frame for an MCF data structure.*

Description

This function is called by generateMCFOneGroup(). However, if the user has a medHistory data frame object, they can call this function explicitly to generate the data structure necessary to run an MCF estimate over those patients.

Usage

```
constructMCF(
  medHistoryDF,
  codetypeVector,
  startDate,
  minRecords = 2,
  returnData = TRUE
)
```

Arguments

medHistoryDF	data frame containing a combination of clinical, referral or therapy.
codetypeVector	vector of desired code types to consider. They are "c" clinica, "r" referral, and "t" therapy.
startDate	start of the cohort observation period. Everyone will be adjusted so records run concurrently i.e., a patient enters 2000 and another 2002, both run for one year. They are counted equally.
minRecords	removes anyone with fewer than the minimum number (2 by default) of medHistoryDF entries.
returnData	default TRUE. Otherwise an MCF object is returned having first perform the MCF estimate with default variance by lawless and Nadeau (1995).

Value

data frame with structured data for an MCF estimate calculation.

Examples

```
medHistoryDF <- constructMedicalHistory(testClinicalDF, NULL, testTherapyDF)
MCF_df <- constructMCF(medHistoryDF,
                       c("c","t"),
                       as.Date("2000-01-01"),
                       2,
                       TRUE)
resultMCF <- reda::mcf(reda::Recur(week, id, No.) ~ 1, data = MCF_df)
```

---

constructMedicalHistory

Combines clinical, referral and therapy records in event date order per patient.

Description

Takes at least one function argument and returns a data frame with the columns patid, eventdate, code, codetype. The column codetype identifies the corresponding code as either 'c' clinical, 'r' referral, or 't' therapy. The input argument data frames can contain any number of columns, however, the columns patid, eventdate and medcode/prodcode must be in that order.

Usage

```
constructMedicalHistory(
  orderedClinicalDF = NULL,
  orderedReferralDF = NULL,
  orderedTherapyDF = NULL
)
```

Arguments

`orderedClinicalDF`
: data frame containing clinical records with at least patid, eventdate and medcode (in that order).

`orderedReferralDF`
: data frame containing referral records with at least patid, eventdate and medcode (in that order).

`orderedTherapyDF`
: data frame containing therapy records with at least patid, eventdate and prodcode (in that order).

Value

data frame with columns patid, eventdate, code, and codetype.

Examples

```
clinicalTherapyDF <- constructMedicalHistory(testClinicalDF,
                                             NULL,
                                             testTherapyDF)
```

---

constructSurvivalTimeline

Construct a survival time data.frame for fixed and time-to-event covariates.

Description

Time event data for cox regression and KM survival curves. Please read the documentation of each function argument carefully; it contains very useful information.

Usage

```
constructSurvivalTimeline(
  medHistoryDF,
  indexVector,
  indexPosition,
  eventVector,
  covariateBooleanList = NULL,
  tdCovariateList = NULL,
  tdCovariateBehaviourVector = NULL,
  obsTime,
  fixedCovariateDF = NULL,
  keepEventlessPatients = TRUE,
  endDate
)
```

Arguments

medHistoryDF	data frame containing 'medical history' (generated using constructMCF) of clinicalDF, referralDF, and/or therapyDF. To increase speed remove all events which are not involved in the cox analysis.
indexVector	A vector of event codes or a single date to indicate when the observation starts. Patient records that do not satisfy this are ignored. If a date, it must be of the Data format, e.g.: 2005-02-24. It must not be of type POSIXct. If multiple index events are found in a patient the first is used.
indexPosition	A string denoting either "FIRST" or "LAST" for an index event before the time-to-event. Useful for "first drug X before event A" and "last drug X before event A".
eventVector	A vector of event codes or a date that indicates the time-to-event. If a date it must be of the Data format, e.g.: 2005-02-24. It must not be of type POSIXct.
covariateBooleanList	A named-list of non-time-dependent covariates (event codes). If the covariate is present then a 1 is assigned to that patid in a column named using the List entry name.
tdCovariateList	A named-list of time-dependent covariants (event codes) e.g., tdCovariateList<-list(analgesic=c(drugcode1,drugcode2,...), Triptan=(drugcode1,drugcode2,...)). The named element is used as the column name in the returning data frame. For every patient without a matching covariate a 0 is supplied. For every patient withh a matching covariate a 1 is initially supplied. If a '+' is specified in the tdCovariateBehaviourVector for a particular covariate, further occurences of a covariate event cause the covariate value to incremement by 1.
tdCovariateBehaviourVector	A vector which is equal in length to tdCovariateList. A '+' indicates that the corresponding tdCovariateList entry increments with each new discovery (a new row). A '-' keeps that tdCovariate value fixed on each new tdCovariate discovery, only allowing for a new row.
obsTime	A number (days) to follow. If the event happens before a status = 1, otherwise a status = 0.
fixedCovariateDF	A data.frame of covariate values by patid. The column names are the covariate types, and the corresponding data.frame element values are the covariate values e.g., data.frame(patid=patid, smoking=smokingStatus, BMI=BMIValue, age=ageAt2017). Patients missing a covariate value are assigned NA.
keepEventlessPatients	If TRUE any patients who don't have a time-to-event event on record are still kept as they might yet experience an event. They will yield a 0 in the status.
endDate	A Date type to indicate the end of the record.

Details

Ensure that there are no code overlaps between different event types in the medHistoryDF. For example, ensure that a set of codes for the event e.g., TIA, are exclusive from a set of codes that indicate indexVector and tdCovariateList. This function will look for the first event that comes after the index date. This will try to run in parallel (numCores/2) unless there are fewer than 100 patients. Please do not overlap codes between the eventcodes for index, covariates and time-to-event (doing so will compromise the results). Patients will be ignored if: (1) they have no index or time-to-event

codes, (2) if all index dates happen after time-to-event dates, (3) a time-to-event happens on the same day as a code for an index date.

Value

A data.frame with columns: patid, time (for non-TD survival), status (0/1), columns per covariates.

Examples

```
medHistoryMainDF <- constructMedicalHistory(NULL,NULL,testTherapyDF)
indexVector <- c(769, 707,83, 1888,49)
indexPosition <- "FIRST"
tdCovariateList <- list(firstCovariate=c(227,78))
tdCovariateBehaviourVector <- c("+")
eventVector <- c(11237,5,26,65)
obsTime <- 365
covariateBooleanList <- NULL
fixedCovariateDF <- NULL
endDate <- as.Date("2018-01-01")
coxDF <- constructSurvivalTimeline(medHistoryMainDF,
                                   indexVector,
                                   indexPosition,
                                   eventVector,
                                   covariateBooleanList=NULL,
                                   tdCovariateList=NULL,
                                   tdCovariateBehaviourVector=NULL,
                                   obsTime,
                                   fixedCovariateDF=NULL,
                                   keepEventlessPatients = TRUE,
                                   endDate = endDate)
```

---

drugListDF *Drug prodcode and description data.*

Description

Drug prodcode and description data stored in an R Dataframe.

Usage

```
data(drugListDF)
```

Format

A Dataframe object with columns: code, description.

Source

This is fabricated data. It takes the likeness of data from https://www.cprd.com/ after permission from ISAC.

References

NA

Examples

```
str(drugListDF)
```

`errorChecker` *Checks whether a function argument is NULL or empty.*

Description

This is for developer use only.

Usage

```
errorChecker(x)
```

Arguments

x	A data.frame, List, Vector, or S3 objects FirstDrugObject and MedHistory.

Value

A boolean TRUE if the argument has size or FALSE if the argument is NULL or empty.

`filterPatientsByDrugMatchingDisease` *Retrieve patient drug prescription records by concurrent medical events.*

Description

Searches a therapy dataset for all drugs (supplied or otherwise taken from those present) that coincide with medical events with or without the consideration of comorbidities which can be adjusted using the severity setting. A therapy data frame containing only those drug prescriptions that satisfy the specified severity.

Usage

```
filterPatientsByDrugMatchingDisease(
  clinicalDF,
  referralDF = NULL,
  therapyDF,
  medcodeList = NULL,
  drugcodeList = NULL,
  severity = 1,
  evidenceDF = NULL
)
```

Arguments

clinicalDF	data frame with at least patid, eventdate, medcode as columns. NULL as default.
referralDF	data frame with at least patid, eventdate, medcode as columns. NULL as default.
therapyDF	data frame with at least patid, eventdate, prodcode as columns. NULL as default.
medcodeList	list or vector of medcodes. NULL as default.
drugcodeList	list or vector of prodcodes. NULL as default.
severity	integer can be either 1 - search for all drugs which coincide with a medcode of interest, or 2 - search for all drug which coincide with a medcode of interest and free from comorbidities on those dates. Currently, severity 2 code is incomplete and will be released in a later version.
evidenceDF	data frame of events medcode/prodcode. Similar to a clinicalDF or therapyDF.

Details

Repeat prescriptions are treated with care. Those drugs prescribed on the date of a clinical event of interest are checked for whether they are either: 0 - a one off 1 - the start of a repeat prescription N - any other number

As we can't directly link a drug to a clinical event there are a number of scenarios to consider. 0 - a one off - then this is always included in severity 1 if it coincides with a medical event and if there are no comorbidities (for severity 2). 1 - the start of a repeat prescription one then needs to walk down the patient record and retain the prescriptions for this series regardless as to whether a clinical event of interest occurs on prescribing days. Repeat prescriptions do not need to occur with a GP visit. If a comorbidity happens on the first prescribing day then the series is discounted (with severity 2). N - any number in the prescribing series. This is not so simple. The patient has come in with a complaint and there is a prescription event for a drug of interest with the same date. However, it's not the first in a sequence of prescriptions (1), indicating that it could have been given for a different complaint earlier and that previously prescribed medication is overlapping with a new condition. However, one must consider the situation when a patient comes in for a routine follow-up to an existing complain and the GP recommends that they continue taking the previously prescribed medication. Therefore, overlaps between condition and prescriptions with a issueseq > 1 are also kept.

The function argument evidenceDF is a data frame of events medcode/prodcode which indicate that any prescriptions of drugs in the drugcodeList that fall on these dates (with these events) concerns the disease of interest. For example, Triptans are only given to patients with a headache. However, headaches can also be treated with preventatives. Unfortunately, preventatives can also prescibed for other conditions. We can make a rough appproximate of the use of these preventatives by not only retaining them according to the list of rules above but also if they are being prescribed on the same date as a Triptan. This is in the instance of when a patient with a history of headache sees their GP. The GP does not record a headache on their record (because they are currently undergoing treatment, it is not a new diagnosis), but the GP begins a prescription for the drug of interest (a migraine preventative) along with the indicator events, the prescription of Triptans.

Value

data frame containing only those patient prescription records that satisfy the search criteria.

Examples

```
requiredProds <- unique(testTherapyDF$prodcode)
allMedCodes <- unique(testClinicalDF$medcode)
```

```
headacheCodes <- allMedCodes[1:10]
mockComorbidityCodes <- allMedCodes[11:52]
fdo <- getFirstDrugPrescription(testTherapyDF,
                                idList=NULL,
                                prodCodesVector=requiredProds,
                                descriptionFile=NULL)
filteredTherapyDF <- filterPatientsByDrugMatchingDisease(testClinicalDF,
                                                         NULL,
                                                         testTherapyDF,
                                                         medcodeList = headacheCodes,
                                                         drugcodeList = requiredProds,
                                                         severity = 1)
```

`generateMCFOneGroup` *Structure data for a single-group Mean cumulative frequency estimate calculation.*

Description

This is a semi-parametric analysis over CPRD clinical, referral or therapy data. It gives an indication of burden/demand on primary care services.

Usage

```
generateMCFOneGroup(
  clinicalDF = NULL,
  referralDF = NULL,
  therapyDF = NULL,
  startDateCharVector = "2000-01-01",
  minRecords = 2
)
```

Arguments

`clinicalDF`	data frame of clinical events. Must contain at the columns "patid", "eventdate", medcode".
`referralDF`	data frame of referral events. Must contain at the columns "patid", "eventdate", medcode".
`therapyDF`	data frame of therapy events. Must contain at the columns "patid", "eventdate", prodcode".
`startDateCharVector`	character vector to denote the date of when to begin including patient events, e.g., "2000-03-21".
`minRecords`	number of records a patient must have to be included in the analysis. Minimum (and default) is 2.

Details

For multiple group member comparisons e.g., for two drugs amitriptyline and topiramate, run this function per drug then perform a cbind to each returning data frame with the name of the drug, then finally, rbind both data frames together. Perform: resultMCF <- mcf(Recur(week, id, No.) ~ Drug, data = drugDF). Default variance by Lawless and Nadeau (1995).

Value

MCF object which can be used to plot. Alternatively, the mean, upper and lower bounds confidence intervals can be extracted and used for bespoke plotting.

Examples

```
numPatients <- length(getUniquePatidList(testTherapyDF))
idList <- getUniquePatidList(testTherapyDF)
maleIDs <- idList[1:1500]
maleClinicalDF <- testClinicalDF[testClinicalDF$patid %in% maleIDs,]
maleTherapyDF <- testTherapyDF[testTherapyDF$patid %in% maleIDs,]
femaleIDs <- idList[1501:length(idList)]
femaleClinicalDF <- testClinicalDF[testClinicalDF$patid %in% femaleIDs,]
femaleTherapyDF <- testTherapyDF[testTherapyDF$patid %in% femaleIDs,]
maleMCFDF <- generateMCFOneGroup(maleClinicalDF,
                                 NULL,
                                 maleTherapyDF)
femaleMCFDF <- generateMCFOneGroup(femaleClinicalDF,
                                   NULL,
                                   femaleTherapyDF)
maleMCFDF <- cbind(maleMCFDF, Gender="Male")
femaleMCFDF <- cbind(femaleMCFDF, Gender="Female")
genderMCF <- rbind(maleMCFDF, femaleMCFDF)
resultMCF <- reda::mcf(reda::Recur(week, id, No.) ~ Gender, data = genderMCF)
```

---

getAgeGroupByEvents *Calculates the age of a patient given their age at a particular year and the date at an event.*

Description

CPRD provides patient age as an adjusted integer that can be decoded given a fixed integer and the year of the CPRD database e.g., 2017. Before using this function ensure that the ageDF argument is a data.frame of unique patid values with the age of the patient calculated from the database release year. Provide that same database release year as the value to the ageAtYear argument.

Usage

```
getAgeGroupByEvents(idList, eventdateList, ageDF, ageGroupVector, ageAtYear)
```

Arguments

idList	List of patid vectors.
eventdateList	List of eventdates. The format of eventdateList is identical to the second element in the FirstDrugObject from getFirstDrugPrescription. The Nth eventdate vector corresponds with the Nth patid vector.
ageDF	data frame with at least columns "patid" and "age." The age is precalculated for a particular year and that year is passed in via ageAtYear.
ageGroupVector	vector of age in years used to denote the lower inclusive bounds of an age group. For example c(18,25,30,...) denotes: 18-24, 25-29, 30... The last age cannot be greater than 199. A patient age during an event less than min(ageGroupVector) is ignored.

ageAtYear — character string or integer denoting a year #### e.g., "2017" or 2017. This will always be converted to a number. If any of the eventdates in eventdateList happens after this number the patient age will be incorrect.

Details

The patids are provided in a list of patid vectors and the eventdates are also provided in a list of eventdate vector. The two lists are identical in List structure and size. The Nth patid vector corresponds with the Nth eventdate vector. The Mth patid in the Nth patid List element vector corresponds to the Mth eventdate in the Nth eventdate List element vector. The format of idList is identical to the first element in the FirstDrugObject from getFirstDrugPrescription. Patids must be unique across the whole list, i.e., a patid can be found once across all List patid Vectors. Duplicate patids will return a NULL.

Value

List of group data.frames. Each element of the list is a data.frame that corresponds to the same list index position in idList and eventdateList. Each column in an age group with the count of events that occured at that age interval.

Examples

```
drugList <- unique(testTherapyDF$prodcode)
requiredProds <- drugList[1:5]
#ensure column names are as follows:
names(ageGenderDF) <- c("patid","age","gender")
ageDF <- ageGenderDF
ageGroupVector <- c(18,25,30,35,40,45,50,55,60)
ageAtYear <- "2017"

resultList <- getFirstDrugPrescription(testTherapyDF,
                                       idList=NULL,
                                       prodCodesVector=requiredProds,
                                       descriptionFile=NULL)
#get the first two vectors of patids (each associated with the drug that can
#be deciphered from the drug code name of each named element in this list).
idList <- as.list(resultList[[1]][1:2])
#get the first two vectors of events dates (corresponds to the first two
#patid vectors above)
eventdateList <- as.list(resultList[[2]][1:2])

ageListResult <- getAgeGroupByEvents(idList,
                                     eventdateList,
                                     ageDF,
                                     ageGroupVector,
                                     ageAtYear)

firstPrescriptionByAgeGroupDF <- ageListResult[[1]]
secondPrescriptionByAgeGroupDF <- ageListResult[[2]]
```

`getBurnInPatients` *Searches for those patients with sufficient record presence for a given time before an event of interest.*

Description

For example, if the start of the burn in is denoted as and the event of interest is and denotes the end of the burn in duration a therapeutic time line may look like: start_of_record|>-x–x–x–x—x–(duration==365 days)–x——x–Y–x->|end_of_record

Usage

```
getBurnInPatients(df, startCodesVector, periodDaysBefore = 365)
```

Arguments

`df`	A data frame with patid column and either medcode (clinical) or prodcode (therapeutic) column.
`startCodesVector`	A vector of codes for the event of interest.
`periodDaysBefore`	An integer of days of burn in time: the duration in days immediately before the event of interest specified in the startCodesVector.

Details

In this patient example, 365 days of time had elapsed between a record of x (where x is a placeholder for any event in the provided data frame with the exception of those code in the startCodesVector) and the event of interest (a record of code Y). Consider this example as a template for all patients with any number of prodcode events and the event of interest being amitriptyline (amitriptyline prodcodes specified in the startCodesVector). This function returns all patients who have therapeutic events occurring at least 365 days before their amitriptlyine prescription.

An example of a patient record without sufficient burn in time:

start_of_record|>————–x–(duration==365 days)–x——x–Y–x->|end_of_record

This patient had no record before the start of the burn in observation window i.e., the duration in days immediately before the event of interest Y.

This is very useful e.g., if one wishes to build a cohort of CPRD patients for interrupted time series analysis. For example, the input therapeutic data frame only contains all triptans (headache analgesics) and the all prescriptions for propranolol (an off label migraine prophylactic). The first prescription of propranolol is the event of interest. Patient ID are collected if there are records of triptans more than 365 days before the first propranolol prescription. One can then use this first-prescription as the point of intervention. Those patients with records that do not extend beyond the 365 observation window are seen as having insufficient record length for pre-intervention regression analysis.

Note: This does not support medcode and prodcode at the same time. Searching for all patients with triptans at least 365 days before a CVD medcode will not work. The user must provide either a clinical data frame or a therapeutic data frame. The code will parse for columns medcode and prodcode and act accordingly. Supporting both is a planned feature.

Value

A list of patids of those patients that satisfy the burn in period. Will return NULL on error.

Examples

```
drugOfInterestVector <- c(83,49,297,1888,940,5)
burnInTime <- 172
#testTherapyDF contains all therapy codes for these patients. Look for those
#patients with at least 172 of event records.
patientList <- getBurnInPatients(testTherapyDF,
                                 drugOfInterestVector,
                                 burnInTime)
```

`getDrugPersistence` *Retrieves data on the proportion of patients stilll taking a particular drug n days later.*

Description

This does not provide information on the events between the first instance of a drug prescription and any instance n days later. It will provide information on the presence of a drug n days later with some degree of buffer prior to that date. This is not for observing a change in medication. The potential first prescribing is testing meds might also be the same type of meds for those e.g., a year later.

Usage

```
getDrugPersistence(
  therapyDF,
  idList = NULL,
  prodcodeList = NULL,
  duration = 365,
  buffer = 30,
  endOfRecordDate = "2017-12-31"
)
```

Arguments

`therapyDF`	A data frame of therapy events.
`idList`	Patients of interest. If left NULL all the patients in therapyDF are used.
`prodcodeList`	A list or vector of prodcodes to identify drugs of interest. If left NULL then all prodcodes in the therapyDF are used.
`duration`	Number of days to check whether the patient was prescribed the same drug they started on.
`buffer`	A number of days from before the duration cut-off that the presence of a drug is valid.
`endOfRecordDate`	A date in the form as.Date("yyyy-mm-dd") .Any patients in which their first drug of interest plus duration time exceeds this date are ignored. This provides an upper boundary to the data set and prevents pulling in patients who are lost to follow up.

Details

Therefore, two individuals would both be considered taking a medication n days later: start–D–D—D—D—D—-D—D–|–D–end start–D————————————D–|–D–end even though the first patient continued taking the drug D from their first prescription up to n days later (with buffer) and the second patient had a significantly long break from the prescription, despite then being prescribed the medication shortly before the end of the period of interest.

This is very useful to simply monitor the proportion of patients still being prescribed a medication n days later whilst being blind to the events that took place in between.

Due to the nature of some datasets, CPRD especially, just observing n days later whether there is an indication for a prescription is very unlikely to yield any interesting results. Rather, a buffer is provided to define a window of time before the duration cut-off to look for records of drugs of interest. For example, given a duration of 365 days and a buffer of 30 day, all prescriptions for e.g., amitriptyline are valid if they fall within 335 - 365 days after their first amitriptyline prescription.

Value

A list. Each element of the list contains a nested list with the elements patid and prodcode. Patid is the patient ID and prodcode is a vector of all prodcodes found within the time of interest.

Examples

```
patientList <- getDrugPersistence(testTherapyDF,
                                  idList=NULL,
                                  prodcodeList=NULL,
                                  duration=395,
                                  buffer=60,
                                  endOfRecordDate="2017-12-31")
```

---

getEventdateSummaryByPatient

Return drug summary data for a single patient.

Description

The following information is returned in a List of two parts. The first element is a vector of the time in days between each drug prescription and the second element is a data frame with the following information (see the example for further information): -patid of patient -number of event dates -median days between events -minimum time between events -maximum time between events -longest duration between events -record duration (from first to last record) - 0 if only one event

Usage

```
getEventdateSummaryByPatient(df, prodCodesVector = NULL)
```

Arguments

`df`	data frame of a patient's therapeutic records.
`prodCodesVector`	vector of product codes to analyse. All other codes in the df are ignored.

Value

S3 EventdateSummaryObj object of patient prescription information.

Examples

```
idList <- getUniquePatidList(testTherapyDF)
resultList <- getEventdateSummaryByPatient(
   testTherapyDF[testTherapyDF$patid==idList[[1]],]
)
str(resultList)
#List of 2
#$ TimeSeriesList: num [1:6] 336 652 2540 34 42 44
#$ SummaryDF     :'data.frame': 1 obs. of  7 variables:
#..$ patid          : int 3101001
#..$ numberOfEvents : int 7
#..$ medianTime     : num 190
#..$ minTime        : num 34
#..$ maxTime        : num 2540
#..$ longestDuration: num 2540
#..$ recordDuration : int 3648
#- attr(*, "class")= chr "EventdateSummaryObj"
```

---

getFirstDrugIncidenceRate

Calculates an incidence rate for prescription in total person-time.

Description

Incidence rate of first prescription allows those enrolled to enter and leave the study at will. Person-time assumes equal probability of incident i.e., 10 people followed for 1 year is the same as 1 person followed for 10 years. Person-time is also known as person-years.

Usage

```
getFirstDrugIncidenceRate(
  firstDrugObject,
  medHistoryDF,
  enrollmentDate,
  studyEndDate
)
```

Arguments

firstDrugObject	Drug first prescription object.
medHistoryDF	All clinical, referral and therapy data available for the cohort. This information must also include the therapy data used to retrieve a first drug prescription. If not, those patients will be removed.
enrollmentDate	Date object of when to study from e.g., as.Date("2000-01-01"). The enrollment is always treated as the start of the year.
studyEndDate	Date object of when the observation ends e.g., as.Date("2010-12-31"). The end date is always treated as the end of the year.

Details

As a good number of patient records will have no information on their entering or leaving the CPRD/electronic healthcare record dataset, we need to use as much of their medical data to determine whether a person will contribute to a particular year. By observing the absence of health indicators, one does not have to provide death-data and one is able to include patients who are lost to followup (LTF) cases when a patient may (a) have transferred to a different medical practice not enrolled on the CPRD scheme or (b) they (in cases of recurrent disease) have decided to give up on treatment.

The accuracy of incidence rate is dependent on the amount of information provided by the user. The precision is down to half person-years.

The observation period is defined from the enrollmentDate through to the the studyEndDate. Please ensure these are year start/end dates to ensure the half-person-year contributions make sense.

Note: Records/indicators/events are used interchangeably, they are synonymous with the eventdate column in the medHistoryDF object (also, a clinical, referral and therapy dataframe).

This function encapsulates the following behaviour:

1) Only those years from the enrollmentDate to the studyEndDate make up the person-years of the population at risk. This is only possible if there are recorded health indicators to parse. A patient must have at least one record to parse, this can be the first drug prescription date if that is their only record.

2) Patients are not included if their drug prescription event happened before the study begins.

3) On entry into a study, a patient will contribute one-full person-year if the first medical indicator on record happened within the first six months (half year) of that study opening year. If the medical indicator on record happened after or on the 1st of July of that opening year, that patient contributes half a person-year. This definition for half a year continues throughout this description.

4) If health indicators on a patient's record stop before the second-half of the last year (e.g., their last indicator is 2010-04-24 and the observation period ends 2010-12-31) of the observation period then that patient is LTF and they contribute half a person-year. All records with a date inside the last half of the final year will contribute a full person-year. Any patient with a single-event can only contribute half a person-year before (a) being LTF as nothing happened after, (b) being LTF as this happened within the last half of the final observation year, or (c) that single event was a first drug prescription and they contributed half a person-year.

5) Contribution to person-years stops when a first drug prescription event is found within the observation period. The position of that drug prescription date is determined, half a person-year is contributed if the prescription date falls within the first half of that year and a full person-year is contributed if the prescription date falls within the second half of that year. Before counting contributions to person-years, the function removes all events after a first drug prescription (if within the observation period). If the drug prescription observation happens after the end of the observation period, it won't count, the algorithm also removes all events after the end of the observation period (for speed).

6) Patients do not need a record for every year from study entry and up to the end of the observation. The first record from entry dictates the behavior of the first contribution of person-year (as described above in (3)), the last record (either LTF, end of study, or first drug prescription) within the observation period dictates the behaviour of the last person-year contribution (as described above in (4)). All years bound between the first valid and last valid indicators are counted as part of the observation study period, and each year contributes to a person-year.

7) If a patient has all record data before the start of the observation period, they will not enter the study (one record must occur within the study period for that patient to contribute). If the first record happens after the end of the observation period, they are also not included.

8) If there are no patients left having first removed those records that are not appropriate for the defined observation period, the algorithm will return NULL. Even if there are many patients in the population-at-risk, if the algorithm at risk has removed everyone with a first drug prescription, then function returns a NULL.

The function will attempt to run in parallel using half the number of available cores.

Value

NULL if the algorithm runs out of patients. Otherwise, a matrix containing patient person-years contribution.

Examples

```
drugList <- unique(testTherapyDF$prodcode)
requiredProds <- drugList[1:10]
#what we have here is a slimmed down data dictionary file - this has a very specific layout
fileLocation <- NULL
firstDrugObject <- getFirstDrugPrescription(testTherapyDF,
                                            idList=NULL,
                                            prodCodesVector=requiredProds,
                                            descriptionFile=fileLocation)
enrollmentDate <- as.Date("2000-01-01")
studyEndDate <- as.Date("2016-12-31")
medhistoryDF <- constructMedicalHistory(testClinicalDF, NULL, testTherapyDF)
patidList <- unlist(firstDrugObject$patidList)
resultMatrix <- getFirstDrugIncidenceRate(firstDrugObject,
                                          medhistoryDF,
                                          enrollmentDate,
                                          studyEndDate)
```

`getFirstDrugPrescription` *Calculates the number of patients prescribed a drug before all other drugs.*

Description

Returns an S3 List object called FirstDrugObject, which contains all the information necessary to characterise the cohort based on first drug prescriptions. By providing a therapy data.frame, which may have already gone through a series of preprocessing steps e.g., having used getPatientsWithFirstDrugWithDisease first, the algorithm will return a linked patid and eventdate for the first drug prescribed on record. The first drug is determined either by (a) what the user provides via a list of prodcodes (any drugs available in the data frame which are not specified by the user are reported in the fourth element of FirstDrugObject as a data frame) or (b) if no drug list is provided by the user the algorithm will use all drugs in the data frame as the search criteria. For each first drug event identified, the patient patid and the event date added to the prodcode-named list entry in patidList (FirstDrugObject[[1]]) and eventdateList (FirstDrugObject[[2]]). For example, for the product Amitriptyline 10mg, the prodcode 83 is used to name the element in each of the two Lists as described. Thus, for each patient with prodcode 83 coming first (before all drugs or those requested), the patient patid and eventdate of drug prescription is added to the vector inside FirstDrugObject[[1]]‘83‘ and FirstDrugObject[[2]]‘83‘.

Usage

```
getFirstDrugPrescription(
  df,
  idList = NULL,
  prodCodesVector = NULL,
  descriptionFile = NULL
)
```

Arguments

df	Therapy data frame having performed some action such as find drug matching with disease.
idList	A list or vector of patient patids to consider. If this is not included all records in the therapy df will be used.
prodCodesVector	A list or vector of prodcodes to consider. If this is not included all available prodcodes in the therapy df will be used.
descriptionFile	A character vector with the file location of the CPRD data dictionary product.txt file. If NULL prodcodes aren't given a description.

Details

The number of first prescriptions for each prodcode is stored in a data frame which can be accessed using FirstDrugObject[[3]]. The columns are named description, code and Frequency. The Frequency matches the length of the vector inside the corresponding prodcode named vector of the patidList of FirstDrugObject[[1]] and eventdateList of FirstDrugObject[[2]].

There are several subtle behaviours of this function one should be aware of. Firstly, if a list of prodcodes are provided, the fourth element of FirstDrugObject will return a description and prodcode for all those drugs identified in the therapy data frame (at any stage in a patient's record) but not part of the user's drug list. This is useful for a very rough description of what types of drugs the cohort are being exposed to. Secondly, it's best to run this function only once and to ensure it's executed over all drugs of interest; take into account any future drugs your study may be considering. If you compare two separate FirstDrugObjects, the first with drug A and the second with drugs A and B, the number of patients taking drug A as a first line of defence may be reduced after introducing drug B. This happens when a patient took drug B before A, but the algorithm only considers drug B when it's included in the drug list search criteria.

Value

An S3 List object. There are four elements to the list. The first, a list of patid vectors. Each patid vector element is named using the prodcode. The second element is a List of Data vectors. Each Date vector element is named using the prodcode. The third element is a data frame containing the prodcode, description and number of patients with that prodcode as a first drug prescription. The length of each patid Vector and Date vector is equal to the corresponding value in the Freq column. The fourth element to the list is a data frame containing the prodcode and description of any drugs not specified by the user and found in the therapy data frame. If the prodCodesVector is left NULL all drugs become part of the search criteria and therefore this element of the S3 List would be NULL.

Examples

```
fileLocation <- NULL
drugList <- unique(testTherapyDF$prodcode)
requiredProds <- drugList[1:10]
df <- getFirstDrugPrescription(testTherapyDF,
                               idList=NULL,
                               prodCodesVector=requiredProds,
                               descriptionFile=fileLocation)
```

`getFirstDrugPrescriptionByYear` *First prescription frequencies per drug by year.*

Description

Note: this function operates over only those patients with a first-drug prescription event. Retrusn a reclassified (S3) List object. Each list element is named using the year values provided (yearVector) and holds a dataframe of drug names and frequencies.

Usage

```
getFirstDrugPrescriptionByYear(firstDrugObject, yearVector)
```

Arguments

`firstDrugObject`	will contain the drug names, number of patients, and the data on which the prescription was recorded.
`yearVector`	vector of years e.g., c(2000,2004,2008) or c(2000:2016).

Value

S3 DrugByYearObj object (list) that holds data frames denoting drug frequency per year requested.

Examples

```
fileLocation <- NULL
drugList <- unique(testTherapyDF$prodcode)
requiredProds <- drugList[1:10]
fdo <- getFirstDrugPrescription(testTherapyDF,
                                idList=NULL,
                                prodCodesVector=requiredProds,
                                descriptionFile=fileLocation)
yearList <- getFirstDrugPrescriptionByYear(fdo, "2001")
```

getGenderOfPatients *Gets all patients by gender.*

Description

The function links those patients specified in idList with any that are present in genderDF. It returns a data frame of patid linked with a gender value. Any patients specified in the idList but absent from the genderDF are ignored. In CPRD, male is 1 and female is 2. By default both genders are parsed, although that can be controlled by used genderCodeVector. Saves you the time with defining an idList for each gender.

Usage

```
getGenderOfPatients(idList, genderDF, genderCodeVector = c(1, 2))
```

Arguments

idList	List of patid values.
genderDF	A data frame with at least "patid" and "gender" columns.
genderCodeVector	An integer vector specifying either 1, 2 or c(1,2) for male, female or male and female.

Value

Data frame with "patid" and "gender". NULL otherwise.

Examples

```
idList <- getUniquePatidList(testTherapyDF)
idList <- idList[1:(length(idList)/2)]
malePatientsDF <- getGenderOfPatients(idList, ageGenderDF, 1)
femalePatientsDF <- getGenderOfPatients(idList, ageGenderDF, 2)
allPatientsDF <- getGenderOfPatients(getUniquePatidList(testTherapyDF),
                                     ageGenderDF)
```

getIMDOfPatients *Get all patients with IMD scores.*

Description

Will return a data frame of patient records linked to an IMD score. Note: approximately 1/4 of patients in the CPRD Gold database have been linked with an IMD score. This function only returns those patients with a linked record.

Usage

```
getIMDOfPatients(idList, imdDF, imdScoreVector = c(1, 2, 3, 4, 5))
```

Arguments

idList	List of patid values. Those patients without a corresponding IMD score are ignored.
imdDF	A data frame with at least "patid" and "score" columns.
imdScoreVector	If supplied, a vector of IMD score to search against. Default are all scores between 1:5.

Value

Data frame with "patid" and "score". NULL otherwise.

Examples

```
idList <- getUniquePatidList(testTherapyDF)
idList <- idList[1:(length(idList)/2)]
onePatientsDF <- getIMDOfPatients(idList, imdDF, 1)
twoPatientsDF <- getIMDOfPatients(idList, imdDF, 2)
allPatientsDF <- getIMDOfPatients(getUniquePatidList(testTherapyDF), imdDF)
```

---

getMultiPrescriptionSameDayPatients

Returns only those patients that do not have multiple prodcodes on the same day.

Description

Returns only those patients that do not have multiple prodcodes on the same day.

Usage

```
getMultiPrescriptionSameDayPatients(
  df,
  prodCodesVector = NULL,
  returnDF = TRUE,
  removePatientsWithoutDrugs = FALSE
)
```

Arguments

df	Data frame with a "patid" column and a "prodcode" column (for a therapy data frame) or a "code" column (for a medHistory data frame).
prodCodesVector	if NULL (default) the function looks at any drug code to determine whether a patient should be excluded for having multiple prescription drugs on the same day. A NULL (default) value is useful if the df passed in has already been cleaned of all but the essential prodcodes.
returnDF	if TRUE return a data frame else return a list of patient patid values.
removePatientsWithoutDrugs	if TRUE will remove all those patients who didn't have those drugs specified in the prodCodesVector.

Value

data frame of patient records or a list of patient patid values.

Examples

```
df <- getMultiPrescriptionSameDayPatients(testTherapyDF)
prodcodesVector <- unique(testTherapyDF$prodcode[1:20])
medHistoryDF <- constructMedicalHistory(NULL,
                                        NULL,
                                        testTherapyDF)
df <- getMultiPrescriptionSameDayPatients(medHistoryDF,
                                          prodcodesVector,
                                          removePatientsWithoutDrugs=TRUE)
```

---

getParallelDF *Not to be called explicitly by package users. Breaks a data frame into a chunk for every computer core.*

Description

This is for multiple-processing. If a dataframe contains 20,000 patient records (thus at least 20,000 rows) and four cores are requested, a list of four dataframes, each containing 5000 patient records is returned.

Usage

```
getParallelDF(df, numCores)
```

Arguments

df	dataframe to divide into chunks.
numCores	number of cores to use.

Details

Unfortunately, package development in line with CRAN requirements limits the number of cores available in a package to 2! I have no idea if this is going to change. For now, I have hard coded a switch to stop R from taking any more than 2 cores. This is such a shame and will slow many of the functions down.

Value

List of dataframes. Each dataframe contains Sum(number_of_patients)/cores, although the last entry is rounded to ensure all patients are included.

Examples

```
numCores <- 4
dfList <- getParallelDF(testTherapyDF, numCores)
```

`getPatientsWithFirstDrugWithDisease` *Searches for those patients with a first-drug matched to disease event date entry.*

Description

This is used to look for all patients where a first drug prescription was prescribed in inline with a disease of interest. This is to ensure that there is no mistaking of what the first drug was prescribed for. Expect a potential big drop in patients in the returning data. For example, some patients might consult a doctor, be told to wait a couple of days then a follow-up call consultation takes places and only a drug is prescribed (rather than a repeat of the original complaint on record). Unfortunately, for now, those patients are ignored. returnIDList = FALSE by default the function will return a modified therapyDF based only on the patients from the clinicalDF. If this is TRUE then the IDlist of those patients will be returned instead. Keeping it set to FALSE is easier to use.

Usage

```
getPatientsWithFirstDrugWithDisease(
  clinicalDF,
  therapyDF,
  medCodesVector = NULL,
  drugCodesVector = NULL,
  returnIDList = FALSE,
  bufferVector = c(0, 0)
)
```

Arguments

`clinicalDF`	data frame of those patients you are interested in. Each patient must have a drug record (entry in therapyDF).
`therapyDF`	data frame of the drugs of interest. Each patient must have a clinical record (entry in clinicalDF).
`medCodesVector`	vector of clinical medical codes to look for when matching first disease with first drug. All other clinical events are removed.
`drugCodesVector`	vector of product codes to look for when matching first disease with first drug. All other therapy events are removed.
`returnIDList`	if TRUE returns an adjusted therapyDF (default) else returns the patient IDs.
`bufferVector`	An integer vector of two values. The first denotes the number of days before a drug event and then second the number of days after the drug event. Diseases within this buffer are considered as being linked. By default this is c(0,0) so only exact dates match.

Value

data frame (default) of therapyDF for those patients with a matching event date on drug-complaint or the idList of those patients.

Examples

```
returnTherapyDF <- getPatientsWithFirstDrugWithDisease(
   clinicalDF=testClinicalDF,
   therapyDF=testTherapyDF,
   medCodesVector=NULL,
   drugCodesVector = NULL,
   FALSE, c(0,0))
```

`getPatientsWithFirstDrugWithNoDisease`

Returns a data frame of therapy records for all those patients who didn't have a drug of interest during their first clinical event of interest.

Description

For example, if headache is the clinical event of interest and migraine preventatives are the drugs of interest, the returned data.frame will only contain therapy patient records where migraine preventatives were not prescribed during the first headache event.

Usage

```
getPatientsWithFirstDrugWithNoDisease(
  clinicalDF = NULL,
  therapyDF = NULL,
  therapyOfDrugOnDiseaseDF = NULL,
  medCodesVector = NULL,
  drugCodesVector = NULL,
  returnIDList = FALSE
)
```

Arguments

`clinicalDF`	data frame of patient clinical data. Every patient must have a clinical and therapy entry. This is not required when a therapyOfDrugOnDiseaseDF data.frame is provided.
`therapyDF`	data frame of therapy data. This is always required.
`therapyOfDrugOnDiseaseDF`	data frame result from getPatientsWithFirstDrugWithDisease(). This must be accompanied by the original input therapyDF of getPatientsWithFirstDrugWithDisease.
`medCodesVector`	vector of clinical medcodes.
`drugCodesVector`	vector of therapy drug codes.
`returnIDList`	if TRUE return patient ids in a List or FALSE (default) return a data frame of therapeutic records.

Details

There are two ways of executing this: (1) From scratch, by providing the clinicalDF and therapyDF and keeping therapyOfDrugDiseaseDF as NULL, or (2) by including the result from getPatientsWithFirstDrugWithDisease in the argument therapyOfDrugOnDiseaseDF along with a patient therapy dataframe, which overrides the first scenario. We recommend running the second scenario to limit user complexity.

Value

data frame of therapeutic records that only include those patients who didn't have a drug of interest during their first clinical data. The data frame will contain all therapy instances for matching patients. If returnIDList is TRUE then the patids are returned as a list.

Examples

```
returnTherapyDF <- getPatientsWithFirstDrugWithDisease(
   clinicalDF=testClinicalDF,
   therapyDF=testTherapyDF,
   medCodesVector=NULL,
   drugCodesVector = NULL,
   FALSE, c(0,0))
df <- getPatientsWithFirstDrugWithNoDisease(
   therapyDF=testTherapyDF,
   therapyOfDrugOnDiseaseDF=returnTherapyDF)
```

---

getPatientsWithProdCode

Gets all patients prescribed a prodcode.

Description

If a data frame is returned (a subset of the function parameter df), the size can be reduced further by requesting that any records for prodcodes not of interest i.e., not within the prodCodesVector, are removed from the data frame. The default behaviour is to keep all drugs on a patients record if that patient has at least one drug prescription matching an element of the prodCodesVector.

Usage

```
getPatientsWithProdCode(
  df,
  prodCodesVector,
  removeExcessDrugs = FALSE,
  returnIDList = FALSE
)
```

Arguments

df	Data frame with patid column present and either a "prodcode" or a "code" column (therapy dataframe or medical history dataframe, respectively).
prodCodesVector	A vector of prodcodes.

removeExcessDrugs
: Logical. If FALSE (default) all prodcodes are retained in a patients record. If TRUE all drugs in that patient that are not part of the prodCodesVector set are removed before returning. For example, if an patient has N prescriptions for any of the prodcodes specified in prodCodesVector, those are kept, whilst any other drugs are removed. See examples.

returnIDList
: Logical. If FALSE (default) a reduced data frame is returned (or NULL). If TRUE a list of patid values are returned.

Details

This function support medical history dataframes by observing the names of the incoming df columns. If "code" is present then it expects a medhical history, on the other hand, if "prodcode" is present then it expects a therapy dataframe.

Value

Data frame of those patients with a prescription for any of the prodcodes in prodCodesVector.

Examples

```
prodCodesVector <- unique(testTherapyDF$prodcode)
reducedProdCodesVector <- prodCodesVector[1:10]
therapyOfInterestDF <- getPatientsWithProdCode(testTherapyDF,
                                               reducedProdCodesVector)
reducedTherapyOfInterestDF <- getPatientsWithProdCode(testTherapyDF,
                                                      reducedProdCodesVector,
                                                      removeExcessDrugs=TRUE)
#the same number of patients are returned, however, fewer drug records are
#returned in reducedTherapyOfInterestDF.
```

---

getPopulationDrugSummary

Returns population level drug summary by calling getEventdateSummaryByPatient for each patient.

Description

Returns population level drug summary by calling getEventdateSummaryByPatient for each patient.

Usage

```
getPopulationDrugSummary(df, prodCodesVector = NULL)
```

Arguments

df
: data frame of therapeutic records. Requires columns "patid" and "prodcode".

prodCodesVector
: vector of prodcodes. If NULL (default), then all prodcode in the therapy df are considered.

Value

S3 PopulationEventdateSummary object (a list) containing patient level EventdateSummaryObj objects extracted into a data frame in the first element and a list of times (in days) between drug prescriptions.

Examples

```
resultList <- getPopulationDrugSummary(testTherapyDF, prodCodesVector=NULL)
str(resultList)
#List of 2
#$ SummaryDF     :'data.frame': 3838 obs. of  7 variables:
#  ..$ patid          : int [1:3838] 3101001 3235001 3333001 3389001 379002
#..$ numberOfEvents : int [1:3838] 7 21 1 1 13 2 15 2 23 79 ...
#..$ medianTime     : num [1:3838] 190 60 0 0 28.5 ...
#..$ minTime        : num [1:3838] 34 34 0 0 11 ...
#..$ maxTime        : num [1:3838] 2540 1623 0 0 322 ...
#..$ longestDuration: num [1:3838] 2540 1623 0 0 322 ...
#..$ recordDuration : num [1:3838] 3648 5329 0 0 630 ...
#$ TimeSeriesList:List of 3838
#..$ 3101001: num [1:6] 336 652 2540 34 42 44
#..$ 3235001: num [1:20] 890 222 182 301 539 ...
#..$ 3333001: num 0
#..$ 3389001: num 0
#..$ 379002 : num [1:12] 26 23 24 24 32 322 31 29 11 51 ...
#..$ 487002 : num 1254
#..$ 637002 : num [1:14] 29 587 9 34 2 0 24 0 22 0 ...
#.. [list output truncated]
#- attr(*, "class")= chr "PopulationEventdateSummary"
```

`getTDTimeline` *Not for external call: builds the cohort level time dependent survival time data through a series of individual patient calls to the corresponding calculateSurvivalTime/calculateTDSurvivalTime.*

Description

Not for external call: builds the cohort level time dependent survival time data through a series of individual patient calls to the corresponding calculateSurvivalTime/calculateTDSurvivalTime.

Usage

```
getTDTimeline(
  medHistoryDF,
  indexVector,
  indexPosition,
  eventVector,
  covariateBooleanList = NULL,
  tdCovariateList,
  tdCovariateBehaviourVector,
  obsTime,
  endDate
)
```

Arguments

medHistoryDF	Dataframe of all necessary medical history for the patients.
indexVector	vector of codes that denote the index date for the patient.
indexPosition	A string denoting either "FIRST" or "LAST" for an index event before the time-to-event. Useful for "first drug X before event A" and "last drug X before event A".
eventVector	vector of codes that indicate an event.
covariateBooleanList	List of named vectors to be used as codes indicating covariates. List element names are used to build data.frame columns.
tdCovariateList	List of covariate vectors. Each covariate vector must be named. e.g., List(nameA=codeVectorA).
tdCovariateBehaviourVector	As with the tdCovariateList but used to denote whether the covariate stays fixed after having been discovered, or does the covariate increment.
obsTime	Number of days to observe events.
endDate	Date indicating when the study ended. Used to determine lost to follow-up patients.

Value

List with the named elements: survivalTime, indexDate and eventDate.

---

getTimeline *Not for external call: builds the cohort level time dependent survival time data through a series of individual patient calls to the corresponding calculateSurvivalTime/calculateTDSurvivalTime.*

Description

Not for external call: builds the cohort level time dependent survival time data through a series of individual patient calls to the corresponding calculateSurvivalTime/calculateTDSurvivalTime.

Usage

```
getTimeline(
  medHistoryDF,
  indexVector,
  indexPosition,
  eventVector,
  covariateBooleanList = NULL,
  obsTime,
  endDate
)
```

Arguments

medHistoryDF	Dataframe of all necessary medical history for the patients.
indexVector	vector of codes that denote the index date for the patient.
indexPosition	A string denoting either "FIRST" or "LAST" for an index event before the time-to-event. Useful for "first drug X before event A" and "last drug X before event A".
eventVector	vector of codes that indicate an event.
covariateBooleanList	List of named vectors to be used as codes indicating covariates. List element names are used to build data.frame columns.
obsTime	Number of days to observe events.
endDate	Date of when the study ends.

Value

List with the named elements: survivalTime, indexDate and eventDate.

---

getUniquePatidList *Retrieves patids from a data frame.*

Description

Retrieves unique patid entries for a health record data frame. If the patid column contains any NAs the code executes but the user is warned. Typically, indication of NAs suggests something went wrong with the creation of the data frame passed to this function.

Usage

```
getUniquePatidList(df)
```

Arguments

df	data frame that contains a patid column.

Value

list of patid numbers. Returns NULL if the data frame is either NULL or empty.

Examples

```
idList <- getUniquePatidList(testClinicalDF)
medHistory <- constructMedicalHistory(testClinicalDF, NULL, testTherapyDF)
idList <- getUniquePatidList(medHistory)
print(paste("There are", length(idList), "patients."))
```

imdDF *Fabricated IMD data.*

Description

Fabricated IMD data stored in an R Dataframe.

Usage

```
data(testTherapyDF)
```

Format

A Dataframe object with columns: patid, pracid, score, As a strict requirement these columns must be included yet the user can add additional columns without concerns over package function compatibility.

Source

This is fabricated data. It takes the likeness of data from https://www.cprd.com/ after permission from ISAC.

References

NA

Examples

```
str(imdDF)
```

loadCPRDDataframe *Load CPRD data frame.*

Description

Use this function to load any data frame including all those generated by this package. Call saveCPRDDataframe() to save a data frame.

Usage

```
loadCPRDDataframe(fileNameString)
```

Arguments

`fileNameString` input file path with file name and extension.

Value

data frame. Returns NULL if the file is not found.

Examples

```
clinicalTherapyDF <- loadCPRDDataframe("my_medHistory.Rda")
```

loadCPRDList *Load CPRD List.*

Description

Use this function to load any base R List including all those generated by this package. Call saveCPRDList() to save a List.

Usage

```
loadCPRDList(fileNameString)
```

Arguments

fileNameString	input file path with file name and extension.

Value

List. Returns NULL if the file is not found.

Examples

```
patidList <- getUniquePatidList(testTherapyDF)
saveCPRDList(patidList, "patientIDs.lst")
patidList <- loadCPRDList("patientIDs.lst")
```

mapDrugTrajectory *Searches for drug prescription switching within a cohort.*

Description

The current implementation does not support multiple processes.

Usage

```
mapDrugTrajectory(
  df,
  clinicalDF = NULL,
  medcodeVector = NULL,
  minDepth = 2,
  maxDepth = 5,
  groupingList = NULL,
  removeUndefinedCode = TRUE
)
```

Arguments

df	data frame of therapy record to perform the drug switching over.
clinicalDF	data frame of clinical record match medcodes with prodcodes. Ensure that for every clinical you have a therapy record.
medcodeVector	A numeric vector of medcodes to apply over the clinicalDF.
minDepth	The minimum number of prescriptions required for a patient to be accepted.
maxDepth	The maximum drug switches considered (including initial i.e., 2 = initial + first switch).
groupingList	A named list of drug code (see example).
removeUndefinedCode	boolean (TRUE by default) indicates whether to remove drug codes not supplied. It will speed up operation if TRUE.

Details

This function returns a number matrix of patients that switch between specified drugs. If you are using clinical data then you must have a therapeutic record for each clinical record.

Value

Currently returns a list of three elements, each is a data frame for plotting. Please see the example for how to plot using the third element off this list.

Examples

```
drugList <- unique(testTherapyDF$prodcode)
requiredProds <- drugList[1:18]
structureList <- list(
  Amitriptyline = c(83,49,1888),
  Propranolol = c(707,297,769),
  Topiramate = c(11237),
  Venlafaxine = c(470,301,39359),
  Lisinopril = c(78,65,277),
  Atenolol = c(5,24,26),
  Candesartan = c(531)
)

resultList <- mapDrugTrajectory(testTherapyDF,
                                NULL,
                                NULL,
                                5,
                                5,
                                groupingList=structureList,
                                removeUndefinedCode=TRUE)
df3 <- resultList[[3]]
#str(df3)
#data.frame': 96 obs. of  6 variables:
#$ FirstDrug: chr  "Amitriptyline" "Propranolol" "Venlafaxine" "Atenolol" ...
#$ Switch1  : chr  "Propranolol" "Amitriptyline" "Propranolol" "Lisinopril" .
#$ Switch2  : chr  "Amitriptyline" "Propranolol" "Venlafaxine" "Atenolol" ...
#$ Switch3  : chr  "Propranolol" "Amitriptyline" "Propranolol" "Lisinopril" .
#$ Switch4  : chr  "Amitriptyline" "Propranolol" "Venlafaxine" "Atenolol" ...
#$ Freq     : num  18 30 5 25 34 6 7 3 1 1 ...
```

```
#required as ggalluvial has not been loaded as a library
StatStratum <- ggalluvial::StatStratum

ggalluvial::is_alluvia_form(as.data.frame(df3), axes = 1:5, silent = TRUE)
ggplot2::ggplot(df3, ggplot2::aes(y = Freq,
        axis1 = FirstDrug,
        axis2 = Switch1,
        axis3 = Switch2,
        axis4 = Switch3,
        axis5 = Switch4)) +
 ggalluvial::geom_alluvium(ggplot2::aes(fill = FirstDrug), width = 1/12) +
 ggalluvial::geom_stratum(width = 1/12, fill = "black", color = "grey") +
 ggplot2::scale_fill_brewer(type = "qual", palette = "Set1") +
 ggplot2::theme_bw() + ggplot2::theme(legend.position = "none") +
 ggplot2::scale_x_discrete(limits = c("First Drug",
                              "1st Switch",
                              "2nd Switch",
                              "3rd Switch",
                              "4th Switch"), expand = c(.05, .05)) +
 ggplot2::ggtitle("Migraine Preventative Switching")
 #include this line if you want labels on the bars
 #ggplot2::geom_label(stat = "stratum", infer.label = TRUE)
```

`matchDrugWithDisease` *Matches a drug with a clinical diagnosis by the same date and returns the list of patient IDs.*

Description

This is extremely useful for generating baseline characteristics of a cohort by answering questions such as, 1) How many patients in the cohort were prescribed drugs x, y, & z? 2) How many patients in the cohort were prescribed drugs x, y, & z at the same time as having a diagnosis for a particular disease (by medcode)? 3) How many patients in the cohort were prescribed drugs x, y, & z at the same time as having an exclusive diagnosis (no accompanying comorbidities on the same date) for a particular disease(by medcode)?

Usage

```
matchDrugWithDisease(
  clinicalDF = NULL,
  referralDF = NULL,
  therapyDF,
  patidList = NULL,
  medcodeList = NULL,
  drugcodeList = NULL,
  severity = 1,
  dateDF = NULL
)
```

Arguments

clinicalDF	data frame with at least patid, eventdate, medcode as columns. NULL as default.
referralDF	data frame with at least patid, eventdate, prodcode as columns. NULL as default.
therapyDF	data frame with at least patid, eventdate, medcode as columns.
patidList	vector of patient IDs. If one hasn't been provided it will be created using the clinicalDF. NULL as default.
medcodeList	vector of comorbidities of interest codes. NULL as default.
drugcodeList	vector of prodcodes of interest. NULL as default.
severity	integer to define search strictness criteria. 1 - a drug anywhere (default), 2 - a drug with a clinical event and comorbidities are not important, 3 - a drug with only a clinical event of interest.
dateDF	data frame with column names patid - start - stop. It must contain all patids available in the combination of clinicalDF, referralDF and therapyDF. NULL as default.

Details

The above three questions are answered by changing the severity parameter. The function returns a list of patient ID (patids) for either further analysis or simply calculating the number of patients by using length(...).

A bounding date interval can be specified. This is a data frame that specifies where to look for each patient in terms of time. By supplying a yearly lower (column name start) and upper (column name stop) date (YYYY-MM-DD) one can determine how many patients were prescribed a particular drug within a given time-frame. The patid can be used to subset the cohort e.g., "all those patients prescribed X drugs with Y condition, who have an IMD of 5 (use the patid in dateDF), between the dates A - B".

Value

A List of patids that are identified to have used a drug of interest.

Examples

```
prodcodes <- unique(testTherapyDF$prodcode)
amitriptylineCodes <- prodcodes[1:5]
propranololCodes <- prodcodes[6:11]

medcodeList <- unique(testClinicalDF$medcode)
headacheCodes <- medcodeList[1:10]
amitriptylineResult1 <- matchDrugWithDisease(clinicalDF = testClinicalDF,
                                             therapyDF = testTherapyDF,
                                             medcodeList = headacheCodes,
                                             drugcodeList=amitriptylineCodes,
                                             severity = 1)
amitriptylineResult2 <- matchDrugWithDisease(clinicalDF = testClinicalDF,
                                             therapyDF = testTherapyDF,
                                             medcodeList = headacheCodes,
                                             drugcodeList=amitriptylineCodes,
                                             severity = 2)
amitriptylineResult3 <- matchDrugWithDisease(clinicalDF = testClinicalDF,
                                             therapyDF = testTherapyDF,
                                             medcodeList = headacheCodes,
```

```
                                          drugcodeList=amitriptylineCodes,
                                          severity = 3)

propranololResult1 <- matchDrugWithDisease(clinicalDF = testClinicalDF,
                                          therapyDF = testTherapyDF,
                                          medcodeList = headacheCodes,
                                          drugcodeList = propranololCodes,
                                          severity = 1)
propranololResult2 <- matchDrugWithDisease(clinicalDF = testClinicalDF,
                                          therapyDF = testTherapyDF,
                                          medcodeList = headacheCodes,
                                          drugcodeList = propranololCodes,
                                          severity = 2)
propranololResult3 <- matchDrugWithDisease(clinicalDF = testClinicalDF,
                                          therapyDF = testTherapyDF,
                                          medcodeList = headacheCodes,
                                          drugcodeList = propranololCodes,
                                          severity = 3)
```

`removePatientsByDuration` *Returns an adjusted therapy data frame containing only those patients who have a minimum observation period.*

Description

This works by searching for events that hold to the minimum number of years for drugs to be present. A further constraint can be added to ensure that a minimum number of observation years is populated by a series of events that are no more than a number of years apart. For example, a record template would look like:

Usage

```
removePatientsByDuration(minObsYr, minBreakYr, therapyDF)
```

Arguments

`minObsYr`	number of years in a drug record to select patients by.
`minBreakYr`	number of years allowed between drug prescriptions before a patient is discounted. If this is set to NA then the minimum number of years is ignored.
`therapyDF`	data frame of clinical or therapeutic patient records.

Value

data frame of those patients with a record that fit the specified criteria.

Examples

```
df <- removePatientsByDuration(5, 2, testTherapyDF)
```

saveCPRDDataframe *Save CPRD data frame.*

Description

Use this function to save any data frame including all those generated by this package. Call loadCPRDDataframe() to load the data frame into the R environment. Save from the working directory set in R.

Usage

```
saveCPRDDataframe(dataFrame, fileNameString)
```

Arguments

dataFrame	data frame to save.
fileNameString	output file path with file name and extension.

Examples

```
clinicalTherapyDF <- constructMedicalHistory(testClinicalDF,
                                             NULL,
                                             testTherapyDF)
saveCPRDDataframe(clinicalTherapyDF, "my_medHistory.Rda")
```

saveCPRDDataframeAsText *Save CPRD data frame as text to file.*

Description

Executes the R write.table(). Very useful for storing text data. Saves to the working directory set in R.

Usage

```
saveCPRDDataframeAsText(df, fileNameString, sepChar = " ")
```

Arguments

df	data frame to save to text file.
fileNameString	output file name and extension.
sepChar	separation between data frame columns. Single white space is default.

Examples

```
clinicalTherapyDF <- constructMedicalHistory(testClinicalDF,
                                             NULL,
                                             testTherapyDF)
saveCPRDDataframeAsText(clinicalTherapyDF, "clinical_therapy_data.txt", " ")
```

saveCPRDList *Save CPRD List.*

Description

Use this function to save any R List object including all those generated by this package.

Usage

```
saveCPRDList(listStr, fileNameString)
```

Arguments

listStr	the List variable.
fileNameString	output file name and extension.

Details

There is no fixed file extension of a list, therefore, we stick to the convention 'lst' so the files become easy to recognise. Call loadCPRDList() to load the List object into the R environment. Saves to the working directory set in R.

Examples

```
clinicalTherapyDF <- constructMedicalHistory(testClinicalDF,
                                             NULL,
                                             testTherapyDF)
patidList <- getUniquePatidList(clinicalTherapyDF)
saveCPRDList(patidList, "patientIDs.lst")
```

sumAcrossAgeGroups *Sums the values by matching each age group with the age group lists generated using getAgeGroupByEvents.*

Description

For example, summing across the matching age groups from the two list elements: [[1]] 18-24 25-29 30-34 35-39 40-44 45-49 50-54 55-59 60+ 104 99 108 134 172 195 175 206 295

Usage

```
sumAcrossAgeGroups(ageGroupList)
```

Arguments

ageGroupList	List generated using getAgeGroupByEvents.

Details

45 41 35 23 44 35 34 20 29

Would yield the named numeric vector result: 18-24 25-29 30-34 35-39 40-44 45-49 50-54 55-59 60+ 149 140 143 157 216 230 209 226 324

Value

numeric vector of each summed age group. Entries are named.

Examples

```
fileLocation <- NULL
drugList <- unique(testTherapyDF$prodcode)
requiredProds <- drugList[1:10]
names(ageGenderDF) <- c("patid","age","gender")
ageGroupVector <- c(18,25,30,35,40,45,50,55,60)
ageAtYear <- "2017"

#get the first drug object
fdoList <- getFirstDrugPrescription(testTherapyDF,
                                    idList=NULL,
                                    prodCodesVector=requiredProds,
                                    descriptionFile=fileLocation)
idList <- as.list(fdoList[[1]][1:2])
eventdateList <- as.list(fdoList[[2]][1:2])

#organise eventdates by age groups
ageListResult1 <- getAgeGroupByEvents(idList,
                                      eventdateList,
                                      ageGenderDF,
                                      ageGroupVector,
                                      ageAtYear)
#sum across the age groups
summedAgeGroupsVector <- sumAcrossAgeGroups(ageListResult1)
```

---

testClinicalDF *Fabricated clinical data.*

Description

Fabricated clinical data stored in an R Dataframe.

Usage

```
data(testClinicalDF)
```

Format

A Dataframe object with columns: patid, eventdate, medcode, consid. As a strict requirement these columns must be included yet the user can add additional columns without concerns over package function compatibility.

Source

This is fabricated data. It takes the likeness of data from https://www.cprd.com/ after permission from ISAC.

References

NA

Examples

```
str(testClinicalDF)
```

`testTherapyDF` *Fabricated therapy data.*

Description

Fabricated therapy data stored in an R Dataframe.

Usage

```
data(testTherapyDF)
```

Format

A Dataframe object with columns: patid, eventdate, prodcode, consid, issueseq. As a strict requirement these columns must be included yet the user can add additional columns without concerns over package function compatibility.

Source

This is fabricated data. It takes the likeness of data from https://www.cprd.com/ after permission from ISAC.

References

NA

Examples

```
str(testTherapyDF)
```

Index