Package: epidm (via r-universe)

October 30, 2024

Version 1.0.5

Title UK Epidemiological Data Management

Description Contains utilities and functions for the cleaning, processing and management of patient level public health data for surveillance and analysis held by the UK Health Security Agency, UKHSA.

URL https://github.com/alexbhatt/epidm, https://alexbhatt.github.io/epidm/

BugReports https://github.com/alexbhatt/epidm/issues

License GPL (>= 3)

Depends R (>= 3.1)

Imports data.table, DBI, odbc, phonics, purrr, readr, stats, stringi, stringr, utils, lubridate

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Repository https://epiverse-connect.r-universe.dev

RemoteUrl https://github.com/alexbhatt/epidm

RemoteRef HEAD

RemoteSha 6fa12b907c618e9cc237634ec974531317f2b712

Contents

`cip_spells` *Continuous Inpatient (CIP) Spells*

Description

[Stable]

A continuous inpatient (CIP) spell is a continuous period of care within the NHS, which does allow specific types of transfers to take place. It can therefore be made up of one or more provider spells. A CIP spell starts when a decision has been made to admit the patient, and a consultant has taken responsibility for their care. The spell ends when the patient dies or is discharged from hospital. This follows the NHS Digital Provider Spells Methodology: http://content.digital.nhs.uk/media/11859/Provider-Spells-Methodology/pdf/Spells_Methodology.pdf

Usage

```
cip_spells(
  x,
  group_vars,
  spell_start_date,
  admission_method,
  admission_source,
  spell_end_date,
  discharge_destination,
  patient_classification,
  .forceCopy = FALSE
)
```

Arguments

x	a data frame; will be converted to a data.table
group_vars	a vector containing any variables to be used for record grouping, minimum is a patient identifier
spell_start_date	Inpatient provider spell or episode admission date
admission_method	CDS admission method code
admission_source	CDS admission source code
spell_end_date	Inpatient provider spell or episode discharge date
discharge_destination	CDS discharge destination code
patient_classification	CDS patient classification code
.forceCopy	default FALSE; TRUE will force data.table to take a copy instead of editing the data without reference

Value

the original data.frame as a data.table with the following new fields:

cip_indx	an id field for the CIP spell
cip_spell_start	the start date for the CIP spell
cip_spell_end	the end date for the CIP spell

Examples

```
cip_test <- data.frame(
  id = c('465','465','465','465','8418','8418','8418',
         '8418','8418','8418','8418','8418','26443',
         '26443','26443','33299','33299','33299','33299',
         '33299','33299','33299','33299','33299','33299',
         '52635','52635','52635','52635','52635','52635',
         '52635','52635','52635','52635','52635','52635',
         '52635','52635','52635','52635','52635','52635',
         '52635','52635','52635','52635','52635','52635',
         '52635','52635','52635','78915','78915','78915'),
  provider = c('X1T','X1T','X1T','X1T','KHA','KHA','KHA',
               'KHA','KHA','KHA','KHA','KHA','BX2','BX2',
               'BX2','PXH','PXH','PXH','PXH','PXH','PXH',
               'PXH','PXH','PXH','PXH','9HA','9HA','9HA',
               '9HA','9HA','9HA','9HA','9HA','9HA','9HA',
               '9HA','9HA','9HA','9HA','9HA','9HA','YYT',
               'YYT','YYT','YYT','YYT','YYT','YYT','YYT',
               'YYT','YYT','YYT','ABX','ABX','ABX'),
  spell_start = as.Date(c(
    '2020-03-07','2020-03-07','2020-03-25','2020-04-03','2020-01-25',
```

```
    '2020-01-26','2020-07-14','2020-08-02','2020-08-12','2020-08-19',
    '2020-08-19','2020-11-19','2019-11-12','2020-04-17','2020-04-23',
    '2020-07-03','2020-01-17','2020-02-07','2020-03-20','2020-04-27',
    '2020-06-21','2020-07-02','2020-10-17','2020-11-27','2021-01-02',
    '2019-12-31','2020-01-02','2020-01-14','2020-01-16','2020-02-07',
    '2020-02-11','2020-02-14','2020-02-18','2020-02-21','2020-02-25',
    '2020-02-28','2020-03-09','2020-03-11','2020-03-12','2020-03-13',
    '2020-03-14','2020-02-04','2020-02-07','2020-02-11','2020-02-14',
    '2020-02-18','2020-02-21','2020-02-25','2020-02-28','2020-03-09',
    '2020-03-11','2020-03-12','2020-04-16','2020-04-24','2020-05-13')),
  spell_end = as.Date(c(
    '2020-03-07','2020-03-25','2020-04-02','2020-04-27','2020-01-25',
    '2020-01-27','2020-07-17','2020-08-07','2020-08-14','2020-08-19',
    '2020-08-22','2020-12-16','2020-04-17','2020-04-23','2020-05-20',
    '2020-07-24','2020-01-28','2020-02-07','2020-03-23','2020-04-29',
    '2020-06-21','2020-07-03','2020-11-27','2021-01-02','2021-01-10',
    '2019-12-31','2020-01-11','2020-01-14','2020-02-04','2020-02-07',
    '2020-02-11','2020-02-14','2020-02-18','2020-02-21','2020-02-25',
    '2020-02-28','2020-03-09','2020-03-11','2020-03-12','2020-03-13',
    '2020-03-30','2020-02-07','2020-02-11','2020-02-14','2020-02-18',
    '2020-02-21','2020-02-25','2020-02-28','2020-03-09','2020-03-11',
    '2020-03-12','2020-03-13','2020-04-24','2020-05-13','2020-06-11')),
  adm_meth = c('21','81','21','81','21','21','11','21','21','21','21',
               '21','21','81','21','81','21','21','21','21','21','21',
               '21','13','13','12','22','12','2D','13','13','13','13',
               '13','13','13','13','13','13','13','21','81','81','81',
               '81','81','13','81','81','13','13','13','21','11','81'),
  adm_src = c('19','51','19','51','19','51','19','51','19','19','19',
              '51','19','51','19','51','19','19','19','19','19','19',
              '19','51','19','19','19','19','19','19','19','19','19',
              '19','19','19','51','51','51','51','19','51','51','51',
              '51','51','51','51','51','51','51','51','19','51','51'),
  dis_meth = c('1','1','1','1','1','1','1','1','1','1','1','1','4','1','1',
               '4','1','1','1','1','1','1','1','8','1','4','1','1','1',
               '1','1','1','1','1','1','1','1','1','1','1','1','1','1',
               '1','1','1','1','1','1','1','1','1','1','1','1','1','2'),
  dis_dest = c('51','51','51','54','51','19','19','19','19','51','19',
               '79','51','51','79','65','19','19','19','19','19','29',
               '98','51','79','19','19','19','51','19','19','19','51',
               '51','51','19','19','51','51','19','51','51','51','51',
               '51','51','51','51','51','51','51','51','29','54','19'),
  patclass = c('1','1','1','1','1','1','1','1','1','1','1','1','1','1','1',
               '1','1','1','1','1','1','1','1','1','1','1','1','2','1','2',
               '1','2','2','2','2','2','2','2','2','2','2','2','1','1',
               '1','1','1','1','1','1','1','1','1','1','1','1','1','1')
)

cip_spells(x=cip_test,
  group_vars = c('id','provider'),
  patient_classification = 'patclass',
  spell_start_date = 'spell_start',
  admission_method = 'adm_meth',
  admission_source = 'adm_src',
```

```
  spell_end_date = 'spell_end',
  discharge_destination = 'dis_dest'
)[]
```

`csv_from_zip` *Download a csv from a zip*

Description

[Stable] A convenience function to allow you to pull data from NHS, ONS and ODR assets

Usage

```
csv_from_zip(x)
```

Arguments

`x`	a zip file from the web

Value

a zip file for ingestion into your chosen readr

Examples

```
## Not run:
read.csv(csv_from_zip("https://files.digital.nhs.uk/assets/ods/current/succarc.zip"))

## End(Not run)
```

`genus_gram_stain` *Bacterial Genus Gram Stain Lookup Table*

Description

A reference table of bacterial gram stain results by genus to allow faster filtering of bacterial results. This dataset has been maintained manually against the PHE SGSS database. If there are organisms missing, please raise and issue or push request on the epidm GitHub

Usage

```
genus_gram_stain
```

Format

A data frame with four columns

organism_genus The bacterial genus

gram_stain A character string to indicate POSITIVE or NEGATIVE type

gram_positive A 0/1 flag to indicate if the genus is gram positive

gram_negative A 0/1 flag to indicate if the genus is gram negative

---

group_ecds_discharge_destination

A&E attendance discharge destination

Description

In order to group A&E discharge destination from SNOWMED into human readable groups, a lookup table has been created. These work with Emergency Care Dataset (ECDS) data with the destination_code field to show where a patient goes after discharge from A&E.

Usage

```
group_ecds_discharge_destination
```

Format

code the ECDS destination_code

destination_code the destination grouping as a human readable string

---

group_inpatient_admission_method

Inpatient admission methods

Description

In order to group hospital inpatient admissions into human readable groups, a lookup table has been created. These work with Hospital Episode Statistics (HES) and Secondary Use Services (SUS) data with the admission_method fields.

Usage

```
group_inpatient_admission_method
```

Format

code the admission_method code

admission_method the admission_method grouping as a human readable string

group_inpatient_discharge_destination *Inpatient discharge destination*

Description

In order to group hospital inpatient discharge destination into human readable groups, a lookup table has been created. These work with Hospital Episode Statistics (HES) and Secondary Use Services (SUS) data with the discharge_destination fields.

Usage

```
group_inpatient_discharge_destination
```

Format

code the discharge_destination code

discharge_destination the discharge_destination grouping as a human readable string

group_time *Grouping of intervals or events in time together*

Description

[Stable]

Group across multiple observations of overlapping time intervals, with defined start and end dates, or events within a static/fixed or rolling window of time. These are commonly used with inpatient HES/SUS data to group spells with defined start and end dates, or to group positive specimen tests, based on specimen dates together into infection episodes.

Usage

```
group_time(
  x,
  date_start,
  date_end,
  window,
  window_type = c("rolling", "static"),
  group_vars,
  indx_varname = "indx",
  min_varname = "date_min",
  max_varname = "date_max",
  .forceCopy = FALSE
)
```

Arguments

x	data frame, this will be converted to a data.table
date_start	column containing the start dates for the grouping, provided quoted
date_end	column containing the end dates for the *interval*, quoted
window	an integer representing a time window in days which will be applied to the start date for grouping *events*
window_type	character, to determine if a 'rolling' or 'static' grouping method should be used when grouping *events*
group_vars	in a vector, the all columns used to group records, quoted
indx_varname	a character string to set variable name for the index column which provides a grouping key; default is indx
min_varname	a character string to set variable name for the time period minimum
max_varname	a character string set variable name for the time period maximum
.forceCopy	default FALSE; TRUE will force data.table to take a copy instead of editing the data without reference

Value

the original data.frame as a data.table with the following new fields:

indx**; renamed using** indx_varname an id field for the new aggregated events/intervals; note that where the date_start is NA, an indx value will also be NA

min_date**; renamed using** min_varname the start date for the aggregated events/intervals

max_date**; renamed using** max_varname the end date for the aggregated events/intervals

Examples

```
episode_test <- structure(
  list(
    pat_id = c(1L, 1L, 1L, 1L, 2L, 2L, 2L,
               1L, 1L, 1L, 1L, 2L, 2L, 2L),
    species = c(rep("E. coli",7),rep("K. pneumonia",7)),
    spec_type = c(rep("Blood",7),rep("Blood",4),rep("Sputum",3)),
    sp_date = structure(c(18262, 18263, 18281, 18282, 18262, 18263, 18281,
                          18265, 18270, 18281, 18283, 18259, 18260, 18281),
                        class = "Date")
  ),
  row.names = c(NA, -14L), class = "data.frame")

group_time(x=episode_test,
           date_start='sp_date',
           window=14,
           window_type = 'static',
           indx_varname = 'static_indx',
           group_vars=c('pat_id','species','spec_type'))[]

spell_test <- data.frame(
```

```
  id = c(rep(99,6),rep(88,4),rep(3,3)),
  provider = c("YXZ",rep("ZXY",5),rep("XYZ",4),rep("YZX",3)),
  spell_start = as.Date(
    c(
      "2020-03-01",
      "2020-07-07",
      "2020-02-08",
      "2020-04-28",
      "2020-03-15",
      "2020-07-01",
      "2020-01-01",
      "2020-01-12",
      "2019-12-25",
      "2020-03-28",
      "2020-01-01",
      rep(NA,2)
    )
  ),
  spell_end = as.Date(
    c(
      "2020-03-10",
      "2020-07-26",
      "2020-05-22",
      "2020-04-30",
      "2020-05-20",
      "2020-07-08",
      "2020-01-23",
      "2020-03-30",
      "2020-01-02",
      "2020-04-20",
      "2020-01-01",
      rep(NA,2)
    )
  )
)

group_time(x = spell_test,
           date_start = 'spell_start',
           date_end = 'spell_end',
           group_vars = c('id','provider'),
           indx_varname = 'spell_id',
           min_varname = 'spell_min_date',
           max_varname = 'spell_max_date')[]
```

`hospital_in_out_dates` *Hospital IN/OUT dates*

Description

This function helps to determine when a patient has been in hospital across spell aggregation. When retaining the final record the following criteria is used:

"1" Current admissions take priority

"2" When conflicting on the same day, inpatient admissions take priority over A&E emergency care data

"3" Where a patient has a linked A&E admission to a hospital inpatient stay, the A&E admission date is used

"4" Where a patient has a positive test between two hospital stays the most recent completed hospital stay prior to the test is retained except if the time between these events is greater than 14 days, then the first admission following the test is retained

Usage

```
hospital_in_out_dates(
  data,
  person_id = "id",
 hospital = list(org_code = "organisation_code_of_provider", event_date = "ev_date",
    ae_arrive = "arrival_date", ae_depart = "departure_date", ae_discharge =
    "ecds_discharge", in_spell_start = "spell_start_date", in_spell_end =
    "spell_end_date", in_discharge = "discharge_destination")
)
```

Arguments

`data`	the linked asset holding A&E and Inpatient data
`person_id`	the column containing the unique patient ID
`hospital`	a list containing the following items
	`org_code` the NHS trust organisation codes
	`event_date` the comparison date used; often `specimen_date`
	`ae_arrive` the ECDS arrival date
	`ae_depart` the ECDS discharge date
	`ae_discharge` the ECDS discharge status; recommend grouping from `epidm::lookup_recode`
	`in_spell_start` the HES/SUS spell start date; recommend after `epidm::group_time`
	`in_spell_end` the HES/SUS spell end date; recommend after `epidm::group_time`
	`in_discharge` the HES/SUS discharge destination code; recommend grouping from `epidm::lookup_recode`

Value

new date columns on the data.table for `hospital_in` and `hospital_out` and `hospital_event_rank`

See Also

epidm::lookup_recode()

epidm::group_time()

epidm::cip_spells()

Examples

```
## Not run:
hospital_in_out_dates(link,
person_id = 'id',
hospital = list(
  org_code = 'organisation_code_of_provider',
  event_date = 'ev_date',
  ae_arrive = 'arrival_date',
  ae_depart = 'departure_date',
  ae_discharge = 'ecds_discharge',
  in_spell_start = 'spell_start_date',
  in_spell_end = 'spell_end_date',
  in_discharge = 'discharge_destination'
))[]

## End(Not run)
```

---

inpatient_codes *Inpatient Codes cleanup*

Description

[Experimental]

When HES/SUS ICD/OPCS codes are provided in wide format you may want to clean them up into long for easier analysis. This function helps by reshaping long as a separate table. Ensuring they're separate allows you to retain source data, and aggregate appropriately later.

Usage

```
inpatient_codes(
  x,
  field_strings,
  patient_id_vars,
  type = c("icd9", "icd10", "opcs"),
  .forceCopy = FALSE
)
```

Arguments

x	a data.frame or data.table containing inpatient data
field_strings	a vector or string containing the regex for the the columns
patient_id_vars	a vector containing colnames used to identify a patient episode or spell
type	a string to denote if the codes are diagnostic or procedural
.forceCopy	default FALSE; TRUE will force data.table to take a copy instead of editing the data without reference

Value

a separate table with codes and id in long form

Examples

```
inpatient_test <- data.frame(
id = c(1053L,5487L,8180L,528L,1085L,344L,2021L,2040L,
       6504L,10867L,12411L,7917L,2950L,2812L,7757L,12227L,2675L,
       8548L,536L,11830L,12708L,10421L,5503L,2494L,14001L),
spell_id = c("dwPDw","iSpUq","qpgk5","8vrJ1","BAur9","l6LZk",
             "KJllb","tgZID","fJkh8","Y9IPv","DAlUZ",
             "9Ooc4","hUxGn","wtMG9","dw3dO","cz3fI",
             "gdxZK","npplb","tynBh","Uu0Sd","gV1Ac",
             "vOpA1","ttlcD","Fqo29","ivTmN"),
primary_diagnosis_code = c("K602","U071-","I501","U071  ","J22X","J189",
                           "J189","I951","N130","U071","K510 D",NA,
                           "G409-","C780","N185","J955","K573","U071",
                           "I330","L309","M513","U071","A419","U071",
                           "N185-"),
secondary_diagnosis_code_1 = c("K641","J128-","I489","J128  ","Q348","F059",
                               "R296","R296","N131","J128","M0750A",NA,
                               "R401-","C782","Z491","C321","D125","J128",
                               "B952","J459","M4780","B972","N390","J128",
                               "Z491-"),
secondary_diagnosis_code_2 = c("E039","B972-","I10X","L031  ","Z115","I509",
                               "F051","I251","K862","B972","K590-",NA,
                               "E876-","C798","N085","Z938","I209","B972",
                               "I214","Z880","M8588","R296","B962","B972",
                               NA),
secondary_diagnosis_code_3 = c("I422","J9691","E119","I489  ","D509","I489",
                               "D509","I252","T391","J440","R21X-",NA,
                               "R945-","E119","M310","I480","I252","J9690",
                               "E111",NA,"Z115","R410","J181","Z518",NA),
secondary_diagnosis_code_4 = c(NA,"I10X-","E669","E109  ","K219","Z921","I251",
                               "I259","R458","B972","F200-",NA,"E039-",
                               "I10X",NA,"I500","F171","I489","E162",NA,
                               "I480","M2551","L892","E86X",NA),
secondary_diagnosis_code_5 = c(NA,"E119-","J449","F03X  ",NA,"Z518","I252",
                               "I209","C61X","A419","R761-",NA,"E119-",
                               "K219",NA,"Z115","F329","N179","N179",NA,
                               "H353","Z638","L033","R54X",NA),
secondary_diagnosis_code_6 = c(NA,NA,"Z966","I10X  ",NA,"N179","N183","Z115",
                               "K627","N390",NA,NA,"J459-","M4780",NA,
                               "Z900",NA,"I10X","R34X",NA,"I951","I10X",
                               "D510","F059",NA),
secondary_diagnosis_code_7 = c(NA,NA,"Z854","I679  ",NA,"N183","Z951","M190",
                               "R634","L031",NA,NA,"I10X-","M512",NA,
                               "Z921",NA,"E119","I959",NA,"H903","I678",
                               "K639","F03X",NA),
secondary_diagnosis_code_8 = c(NA,NA,"Z864","J459  ",NA,"E115","E119","N183",
                               "E111","E871",NA,NA,"R51X-","H409",NA,
                               "Z870",NA,NA,"J90X",NA,"M199","J459",
```

```
                                "N133","F29X",NA),
    secondary_diagnosis_code_9 = c(NA,NA,"Z921","R296  ",NA,"L97X","I10X","M4806",
                                "E114","S099",NA,NA,"Q070-","H544",NA,
                                NA,NA,NA,"I501",NA,"K811","F03X","J90X",
                                "N189",NA),
    secondary_diagnosis_code_10 = c(NA,NA,NA,"Z921  ",NA,"L089","Z921","N40X",
                                 "G590","R296",NA,NA,"E668-","Z858",NA,NA,NA,
                                 NA,"I489",NA,"K219","G20X","N202",
                                 "F719",NA),
    secondary_diagnosis_code_11 = c(NA,NA,NA,"Z515  ",NA,"R02X","Z507","Z864",
                                 "E162","I489",NA,NA,"G473-","Z923",NA,NA,NA,
                                 NA,"I447",NA,"J459","E119","L031",
                                 "Z960",NA),
    secondary_diagnosis_code_12 = c(NA,NA,NA,"Z501  ",NA,"B370","K579","Z955",
                                 "E46X","Z921",NA,NA,"R600-","Z926",NA,NA,NA,
                                 NA,"E86X",NA,"I10X",NA,"J981","Z922",
                                 NA),
    secondary_diagnosis_code_13 = c(NA,NA,NA,"Z507  ",NA,"E039","M109",NA,"I259",
                                 "K709",NA,NA,"M1999","Z895",NA,NA,NA,NA,
                                 "R33X",NA,"J40X",NA,"E119",NA,NA),
    secondary_diagnosis_code_14 = c(NA,NA,NA,NA,NA,NA,"J459",NA,"N131","Z864",NA,
                                 NA,"R468-","Z902",NA,NA,NA,NA,"R296",
                                 NA,NA,NA,"I739",NA,NA),
    secondary_diagnosis_code_15 = c(NA,NA,NA,NA,NA,NA,"Z880",NA,"K862","Z501",NA,
                                 NA,"Z115-","Z971",NA,NA,NA,NA,"R468",
                                 NA,NA,NA,"N183",NA,NA),
    secondary_diagnosis_code_16 = c(NA,NA,NA,NA,NA,NA,"Z867",NA,"T391","Z505",NA,
                                 NA,"Z501-","Z878",NA,NA,NA,NA,"R31X",
                                 NA,NA,NA,"I489",NA,NA),
    secondary_diagnosis_code_17 = c(NA,NA,NA,NA,NA,NA,"Z864",NA,"R458","Z518",NA,
                                 NA,"Z507-","Z958",NA,NA,NA,NA,"Z115",
                                 NA,NA,NA,"M549",NA,NA),
    secondary_diagnosis_code_18 = c(NA,NA,NA,NA,NA,NA,"F03X",NA,"C61X",NA,NA,NA,
                                 NA,"Z867",NA,NA,NA,NA,"I252",NA,NA,
                                 NA,"I252",NA,NA),
    secondary_diagnosis_code_19 = c(NA,NA,NA,NA,NA,NA,NA,NA,"K627",NA,NA,NA,NA,
                                 "Z864",NA,NA,NA,NA,"I259",NA,NA,NA,
                                 "I259",NA,NA),
    secondary_diagnosis_code_20 = c(NA,NA,NA,NA,NA,NA,NA,NA,"R634",NA,NA,NA,NA,
                                 "Z880",NA,NA,NA,NA,"I10X",NA,NA,NA,
                                 "E669",NA,NA),
    secondary_diagnosis_code_21 = c(NA,NA,NA,NA,NA,NA,NA,NA,"E111",NA,NA,NA,NA,
                                 "Z800",NA,NA,NA,NA,"I352",NA,NA,NA,
                                 "Z867",NA,NA),
    secondary_diagnosis_code_22 = c(NA,NA,NA,NA,NA,NA,NA,NA,"E114",NA,NA,NA,NA,
                                 "Z801",NA,NA,NA,NA,"R15X",NA,NA,NA,
                                 "Z896",NA,NA),
    secondary_diagnosis_code_23 = c(NA,NA,NA,NA,NA,NA,NA,NA,"G590",NA,NA,NA,NA,
                                 NA,NA,NA,NA,NA,"R32X",NA,NA,NA,
                                 "Z960",NA,NA),
    secondary_diagnosis_code_24 = c(NA,NA,NA,NA,NA,NA,NA,NA,"E162",NA,NA,NA,NA,
                                 NA,NA,NA,NA,NA,"R418",NA,NA,NA,
                                 "Z874",NA,NA),
```

```
primary_procedure_code = c("H289",NA,"K634",NA,"X292",NA,NA,NA,NA,NA,
                           "H251",NA,"U051","L913","X403",NA,"H231",
                           "U071","M473","X384",NA,NA,NA,NA,"X403"),
primary_procedure_date = c("20170730",NA,"20201202",NA,"20170914",NA,NA,NA,
                           NA,NA,"20210105",NA,"20170724",
                           "20210111","20171114",NA,"20170622","20210104",
                           "20171013","20170313",NA,NA,NA,NA,
                           "20171107"),
secondary_procedure_code_1 = c("H626",NA,"Y534",NA,"U297",NA,NA,NA,NA,NA,
                               "Z286",NA,"Y981","Y031",NA,NA,"Z286",
                               "Y981",NA,NA,NA,NA,NA,NA,NA),
secondary_procedure_date_1 = c("20170730",NA,"20201202",NA,"20170928",NA,NA,NA,
                               NA,NA,"20210105",NA,"20170724",
                               "20210111",NA,NA,"20170622","20210104",NA,NA,NA,
                               NA,NA,NA,NA),
secondary_procedure_code_2 = c("H444",NA,"Z941",NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               "U212",NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               NA,NA,NA),
secondary_procedure_date_2 = c("20170730",NA,"20201202",NA,NA,NA,NA,NA,NA,NA,
                               NA,NA,"20170729",NA,NA,NA,NA,NA,NA,
                               NA,NA,NA,NA,NA,NA),
secondary_procedure_code_3 = c(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,"Y973",
                               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               NA),
secondary_procedure_date_3 = c(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               "20170729",NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               NA,NA),
secondary_procedure_code_4 = c(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,"Y982",
                               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               NA),
secondary_procedure_date_4 = c(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               "20170729",NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               NA,NA),
secondary_procedure_code_5 = c(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,"Z926",
                               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               NA),
secondary_procedure_date_5 = c(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               "20170729",NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               NA,NA),
secondary_procedure_code_6 = c(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,"O161",
                               NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               NA),
secondary_procedure_date_6 = c(NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               "20170729",NA,NA,NA,NA,NA,NA,NA,NA,NA,
                               NA,NA)
)

inpatient_codes(x=inpatient_test,
                field_strings='diagnosis',
                patient_id_vars = c('id','spell_id'),
                type = 'icd10')

inpatient_codes(x=inpatient_test,
```

```
                field_strings=c('procedure_code','procedure_date'),
                patient_id_vars = c('id','spell_id'),
                type = 'opcs')
```

---

lab_data *Synthetic Lab Data for epidm*

Description

A dataset containing synthetic lab data for testing epidemiological data transformation functions.

Usage

```
data(lab_data)
```

Format

A data frame with the following columns:

nhs_number NHS number

local_patient_identifier Patient identifier such as hospital number

patient_birth_date Date of birth of the patients.

sex Gender of the patients (Factor with levels: "Female", "Male").

surname Patient surname

forename Patient forename

organism_species_name Organism species name (Factor with levels: "KLEBSIELLA PNEUMONIAE").

specimen_date Date of specimen collection.

specimen_type Type of specimen: BLOOD or URINE.

lab_code Laboratory codes (Factor with unique levels).

local_authority_name Name of the local authority.

local_authority_code Code of the local authority.

postcode Postcode

Examples

```
data(lab_data)
head(lab_data)
```

lookup_recode *Lookup table switch handler*

Description

[Stable] A function to call an epidm lookup table and recode where we are aware of a new value. Built in are the organism re-classifications and specimen_type groupings and a manual mode.

Usage

```
lookup_recode(
  src,
  type = c("species", "specimen", "inpatient_admission_method",
    "inpatient_discharge_destination", "ecds_destination_code", "manual"),
  .import = NULL
)
```

Arguments

src	a character, vector or column containing the value(s) to be referenced
type	a character value to denote the lookup table used
.import	a list in the order list(new,old) containing the values for another lookup table existing in the environment

Value

a list object of the recoded field

Examples

```
df <- data.frame(
  spec = c(
    sample(grep(")",
                respeciate_organism$previous_organism_name,
                value=TRUE,
                invert = TRUE),
           9),
    "ESCHERICHIA COLI","SARS-COV-2","CANDIDA AUREUS"),
  type = sample(specimen_type_grouping$specimen_type,12),
  date = sample(seq.Date(from = Sys.Date()-365,
                         to = Sys.Date(),
                         by = "day"),12)
)
df <- df[order(df$date),]

# show the data before the changes
df
```

```
# check the lookup tables
# observe the changes
head(respeciate_organism[1:2])
df$species <- lookup_recode(df$spec,'species')
df[,c('spec','species','date')]

head(specimen_type_grouping)
df$grp <- lookup_recode(df$type,'specimen')
df[,c('species','type','grp','date')]

# for a tidyverse use
# df %>% mutate(spec=lookup_recode(spec,'species))

# manual input of your own lookup
# .import=list(new,old)
lookup_recode(
  "ALCALIGENES DENITRIFICANS",
  type = 'manual',
  .import=list(respeciate_organism$organism_species_name,
               respeciate_organism$previous_organism_name)
  )
```

`proxy_episode_dates` *HES/SUS Episode Date Cleaning*

Description

[Stable]

Correcting for missing end dates on HES/SUS episodes

Usage

```
proxy_episode_dates(
  x,
  group_vars,
  spell_start_date,
  spell_end_date,
  discharge_destination,
  .dropTmp = TRUE,
  .forceCopy = FALSE
)
```

Arguments

`x`	a data frame; will be converted to a data.table
`group_vars`	a vector containing any variables to be used for record grouping, minimum is a patient identifier

spell_start_date
: Inpatient provider spell or episode admission date

spell_end_date
: Inpatient provider spell or episode discharge date

discharge_destination
: CDS discharge destination code

.dropTmp
: default TRUE; a logical to drop all tmp values used

.forceCopy
: default FALSE; TRUE will force data.table to take a copy instead of editing the data without reference

Value

a data.table with cleaned start and end dates, and an indicator proxy_missing where the value has changed

Examples

```
proxy_test <- data.frame(
  id = c(
    rep(3051, 4),
    rep(7835,3),
    rep(9891,3),
    rep(1236,3)
  ),
  provider = c(
    rep("QKJ", 4),
    rep("JSD",3),
    rep("YJG",3),
    rep("LJG",3)
  ),
  spell_start = as.Date(c(
    "2020-07-03", "2020-07-14", "2020-07-23", "2020-08-05",
    "2020-11-01", "2020-11-13", "2020-12-01",
    "2020-03-28", "2020-04-06", "2020-04-09",
    "2020-10-06", "2020-11-05", "2020-12-25"
  )),
  spell_end = as.Date(c(
    "2020-07-11", "2020-07-22", "2020-07-30", "2020-07-30",
    "2020-11-11", NA, "2020-12-03",
    "2020-03-28", NA, "2020-04-09",
    "2020-10-06", "2020-11-05", NA
  )),
  disdest = c(
    19, 19, 51, 19,
    19, 19, 19,
    51, 98, 19,
    19, 19, 98
  )
)


proxy_episode_dates(
```

```
  x=proxy_test,
  group_vars = c('id','provider'),
  spell_start_date = 'spell_start',
  spell_end_date = 'spell_end',
  discharge_destination = 'disdest'
)[]
```

`respeciate_generic` *Respeciate unspecified samples*

Description

[Stable]

Some samples within SGSS are submitted by laboratories as "GENUS SP" or "GENUS UNNAMED". However, they may also have a fully identified sample taken from the same site within a recent time period. This function captures species_col from another sample within X-days of an unspeciated isolate.

Usage

```
respeciate_generic(
  x,
  group_vars,
  species_col,
  date_col,
  window = c(0:Inf),
  .forceCopy = FALSE
)
```

Arguments

`x`	a data.frame or data.table object
`group_vars`	the minimum grouping set of variables for like samples in a character vector; suggest c('patient_id','specimen_type','genus')
`species_col`	a character containing the column with the organism species_col name
`date_col`	a character containing the column with the specimen/sample date_col
`window`	an integer representing the number of days for which you will allow a sample to be respeciated
`.forceCopy`	default FALSE; TRUE will force data.table to take a copy instead of editing the data without reference

Value

a data.table with a recharacterised `species_col` column

Examples

```
df <- data.frame(
ptid = c(round(runif(25,1,5))),
spec = sample(c("KLEBSIELLA SP",
                "KLEBSIELLA UNNAMED",
                "KLEBSIELLA PNEUMONIAE",
                "KLEBEIELLA OXYTOCA"),
              25,replace = TRUE),
type = "BLOOD",
specdate = sample(seq.Date(Sys.Date()-21,Sys.Date(),"day"),25,replace = TRUE)
)

respeciate_generic(x=df,
                   group_vars=c('ptid','type'),
                   species_col='spec',
                   date_col='specdate',
                   window = 14)[]
```

---

respeciate_organism *Respeciated organisms*

Description

Occasionally, research shows that two organisms, previously thought to be different are in fact one and the same. The reverse is also true. This is a manually updated list. If there are organisms missing, or new respeciates to be added, please raise and issue or push request on the epidm GitHub

Usage

```
respeciate_organism
```

Format

previous_organism_name What the organism used to be known as, in the form GENUS SPECIES

organism_species_name What the organism is known as now, in the form GENUS SPECIES

organism_genus_name The genus of the recoded organism

genus_change A 0/1 flag to indicate if the genus has changed

genu_all_species A 0/1 flag to indicate if all species under that genus should change

specimen_type_grouping *Specimen type grouping*

Description

In order to help clean up an analysis based on a group of specimen types, a lookup table has been created to help group sampling sites. This is a manually updated list. If there are organisms missing, or new respeciates to be added, please raise and issue or push request on the epidm GitHub

Usage

```
specimen_type_grouping
```

Format

specimen_type The primary specimen type with detail

specimen_group A simple grouping of like specimen sites

sql_clean *Clean and Read a SQL query*

Description

[Stable]

A utility function to read in a SQL query from a character object, clipboard or text file and remove all comments for use with database query packages

Usage

```
sql_clean(sql)
```

Arguments

sql	a SQL file or text string

Value

a cleaned SQL query without comments as a character string

Examples

```
testSQL <- c(
"/********* INTRO HEADER COMMENTS",
"*********/",
"  SELECT ",
"  [VAR 1]  -- with comments",
",[VAR 2]",",[VAR 3]",
"FROM DATASET ","-- output here")
sql_clean(testSQL)
```

`sql_connect` *Connect to a SQL database*

Description

[Stable]

An function to help setup connections to SQL databases acting as a wrapper for the odbc and DBI packages. Used by other sql_* tools within epidm. This uses the credential manager within the system and assumes you are using a trusted connection.

Usage

```
sql_connect(server, database)
```

Arguments

`server`	a string containing the server connection; note that servers may require the use of double backslash \\
`database`	a string containing the database name within the data store

Value

a SQL connection object

See Also

sql_clean sql_read sql_write

Examples

```
## Not run:
sql <- list(
  dsn = list(ser = 'covid.ukhsa.gov.uk',
             dbn = 'infections')
)
```

```
sgss_con = sql_connect(server = sql$dsn$ser, database = sql$dsn$dbn)

## End(Not run)
```

`sql_read` *Read a table from a SQL database*

Description

[Stable]

Read a table object to a SQL database. Acts a wrapper for odbc and DBI packages.

Usage

```
sql_read(server, database, sql)
```

Arguments

`server`	a string containing the server connection
`database`	a string containing the database name within the data store
`sql`	a string containing a SQL query or to a .sql/.txt SQL query

Value

a table from a SQL database

See Also

sql_clean sql_connect

`sql_write` *Write a table to a SQL database*

Description

[Stable]

Write a table object to a SQL database. Acts a wrapper for odbc and DBI packages with additional checks to ensure upload completes.

Usage

```
sql_write(x, server, database, tablename)
```

Arguments

x	a data.frame/data.table/tibble object
server	a string containing the server connection
database	a string containing the database name within the data store
tablename	a string containing the chosen SQL database table name

Value

writes a data.frame/data.table/tibble to a SQL database

---

uk_patient_id *Patient ID record grouping*

Description

[Stable]

Groups patient records from multiple isolates with a single integer patientID by grouping patient identifiers.

Grouping is based on the following stages:

1. matching nhs number and date of birth
2. Hospital number & Date of Birth
3. NHS number & Hospital Number
4. NHS number & Name
5. Hospital number & Name
6. Sex & Date of Birth & Surname
7. Sex & Date of Birth & Fuzzy Name
8. Sex & Year and Month of Birth & Fuzzy Name
9. Postcode & Name
10. Name Swaps (when first and last name are the wrong way around)

Identifiers are copied over where they are missing or invalid to the grouped records.

Usage

```
uk_patient_id(
  data,
  id = list(nhs_number = "nhs_number", hospital_number = "patient_hospital_number",
   date_of_birth = "date_of_birth", sex_mfu = "sex", forename = "forename", surname =
     "surname", postcode = "postcode"),
  .useStages = c(1:11),
  .sortOrder,
  .keepValidNHS = FALSE,
  .forceCopy = FALSE
)
```

Arguments

`data`	a data.frame or data.table containing the patient data
`id`	a named list to provide the column names with identifiers, quoted `nhs_number` the patient NHS number `hospital_number` the patient Hospital numbers also known as the local patient identifier `date_of_birth` the patient date of birth `sex_mfu` the patient sex or gender field as Male/Female/Unknown `forename` the patient forename `surname` the patient surname `postcode` the patient postcode
`.useStages`	optional, default 1:11; set to 1 if you wish patient ID to be assigned cases with the same DOB and NHS number, set to 2 if you wish patient ID to be assigned to cases with the same hospital number (HOS) and DOB, set to 3 if you wish patient ID to be assigned cases with the same NHS and HOS number, set to 4 if you wish patient ID to be assigned cases with the same NHS number and surname, set to 5 if you wish patient ID to be assigned cases with the same hospital number and surname, set to 6 if you wish patient ID to be assigned cases with the same DOB and surname, set to 7 if you wish patient ID to be assigned cases with the same sex and full name, set to 8 if you wish patient ID to be assigned cases with the same sex, DOB and fuzzy name, set to 9 if you wish patient ID to be assigned cases with the same DOB and fuzzy name, set to 10 if you wish patient ID to be assigned cases with the same name and postcode, set to 11 if you wish patient ID to be assigned cases with the same first name or second name in changing order and date of birth.
`.sortOrder`	optional; a column as a character to allow a sorting order on the id generation
`.keepValidNHS`	optional, default FALSE; set TRUE if you wish to retain the column with the NHS checksum result stored as a BOOLEAN
`.forceCopy`	optional, default FALSE; TRUE will force data.table to take a copy instead of editing the data without reference

Value

A dataframe with one new variable:

`id` a unique patient id

`valid_nhs` if retained using argument `.keepValidNHS=TRUE`, a BOOLEAN containing the result of the NHS checksum validation

Examples

```
uk_patient_id(
 data = head(epidm::lab_data),
 id = list(
   nhs_number = 'nhs_number',
   hospital_number = 'local_patient_identifier',
```

```
    date_of_birth = 'patient_birth_date',
    sex_mfu = 'sex',
    forename = 'forename',
    surname = 'surname'
    postcode = 'postcode'
  ),
  .sortOrder = 'specimen_date',
  .forceCopy = TRUE
)[]
```

`valid_nhs` *NHS Number Validity Check*

Description

[Stable]

Check if NHS numbers are valid based on the checksum algorithm

This uses the first 9 digits, multiplied by 10 down to 2 eg digit 1x10, d2x9

The sum of the products of the first 9 digits are divided by 11

The remainder is checked against the 10th digit

Where the remainder is 11, it is replaced with 0

Usage

```
valid_nhs(nhs_number)
```

Arguments

`nhs_number`	a vector

Value

a vector, 1 if NHS number is valid, 0 if not valid

Examples

```
test <- floor(runif(1000,1000000000,9999999999))
valid_nhs(test)
valid_nhs(9434765919)
```

Index