HIV Publications by country

Andreas Ronit and Magnus Glindvad Ahlström

8 maj 2017

medline

This function converts medline files from text format to a meaningful list of variables. For more information: https://mlbernauer.wordpress.com/2014/12/15/parsing-medline-files-in-r/

```
medline <- function(file_name){
  lines <- readLines(file_name)
  medline_records <- list()
  key <- 0
  record <- 0
  for(line in lines){
    header <- sub(" {1,20}", "", substring(line, 1, 4))
    value <- sub("^.{6}", "", line)
    if(header == "" & value == ""){
      next
    }
    else if(header == "PMID"){
      record = record + 1
      medline_records[[record]] <- list()
      medline_records[[record]][header] <- value
    }
    else if(header == "" & value != ""){
      medline_records[[record]][key] <- paste(medline_records[[record]][key], valu
e)
    }
    else{
      key <- header
      if(is.null(medline_records[[record]][key][[1]])){
        medline_records[[record]][key] <- value
      }
      else {
        medline_records[[record]][key] <- paste(medline_records[[record]][key], va
lue, sep=";")
      }
    }
  }
  return(medline_records)
}
```

Functions for data extraction

First we provide some useful functions for MEDLINE data extraction:

findCountry

findCountry extracts country from author affiliation lists. The functions is based on the function countrycode, from the R package 'countrycode'. In short, the function returns a country of the last/senior author of each medline record.

```r
findCountry <- function(medline_record) {
  if("AD" %in% names(medline_record)) {
    str <- strsplit(medline_record$AD, ";")[[1]]
    len <- length(str)
    str2 <- strsplit(str[len], "\\ ")[[1]]
    countries <- suppressWarnings(countrycode(str2,
                                              origin="country.name",
                                              destination = "country.name.en"))
    countries <- countries[!is.na(countries)]
    if(identical(countries, character(0))) {
      str2 <- strsplit(str[len], ",")[[1]]
      countries <- suppressWarnings(countrycode(str2,
                                                origin="country.name",
                                                destination = "country.name.en"))
      countries <- countries[!is.na(countries)]
    }
    countries[!duplicated(countries)]
  } else {
    "No author aff"
  }
}
```

country.by.city

A complete list of cities of the world can be found at: https://www.maxmind.com/en/free-world-cities-database. This function maps cities to country via the above list.

```r
trim.punct <- function(x) gsub("(^[[:punct:]]+|[[:punct:]]+$)", "", x)
country.by.city <- function(city) {
  city <- tolower(city)
  city <- trimws(city)
  city <- trim.punct(city)
  country <- world.cities[world.cities$City %in% city, "Country"]
  if(!identical(country, character(0))) return(country) else NA
}
```

findCountry2 and findCountry3

findCountry2 extracts country from author affiliation lists via cities reported. The function uses the above country.by.city function to match cities with countries. If the city of the last author returns more than on country, the function will search through the affiliations of all other authors and return the country with the most hits. findCountry3 is a wrapper around findContry2 to deal with errors.

```
findCountry2 <- function(medline_record) {
  country.name <- tryCatch(countrycode(medline_record$AD,
                                       origin="country.name",
                                       destination = "country.name.en"),
                           error = function(x) return("Unavailable city data"))
  if(is.na(country.name)) {
    str <- strsplit(medline_record$AD, ";")[[1]]
    len <- length(str)
    str2 <- strsplit(str[len], ",")[[1]]
    country.iso2c <- lapply(str2, country.by.city)
    country.iso2c <- unlist(country.iso2c[!is.na(country.iso2c)])
    country.iso2c <- country.iso2c[!duplicated(country.iso2c)]
    if(is.null(country.iso2c)) {
      return("Unavailable city data")
    } else {
      country.name <- countrycode(country.iso2c, origin="iso2c", destination = "co
untry.name.en")
    }
  }
  if(length(country.name) > 1) {
    str <- strsplit(medline_record$AD, ";")[[1]]
    str0.1 <- str[!unname(sapply(str, function(x) x == ""))]
    str2 <- sapply(str0.1, strsplit, ";")
    str3 <- sapply(str2, strsplit, ",")
    str4 <- sapply(str3, country.by.city)
    str5 <- lapply(str4, function(x) x[!duplicated(x)])
    str6 <- unname(do.call(c, str5))
    str7 <- names(table(str6)[which(table(str6) == max(table(str6)))])
    country.name <- countrycode(str7, origin = "iso2c", destination = "country.nam
e.en")
  }
  return(country.name)
}

findCountry3 <- function(x) {
  tryCatch(findCountry2(x), error = function(x) "An error occurred")
}
```

findDate

findDate extracts the puclication date of the study, and returns the year of publication.

```r
findDate <- function(medline_record) {
  if("DP" %in% names(medline_record)) {
    year.str <- substring(medline_record$DP, 1,4)
    as.numeric(year.str)
  } else {
    "No date"
  }
}
```

The script

This script will extract the data from a medline file and create the plots. You should change the working directory to your own and and make sure you have all relevant data available, i.e. worldcitiespop.txt and the PubMed MEDLINE .txt file. We have provided the lists 'countries', 'all.new.countries', and 'dates' as .Rdata files to avoid running the time consuming parts of the script. These files can be found at https://github.com/magnusahlstroem/Publication_by_country. To download a MEDLINE file go to PubMed and type relevant key words, e.g. HIV [MeSH] AND AIDS [MeSH]. Export these data using Send to > File > MEDLINE.

```r
require(ggplot2)
require(tidyr)
require(dplyr)
require(countrycode)
require(rworldmap)
setwd("H:/Dokumenter/Forskning/Studier/Publication_by_country")

medline_records <- medline("pubmed_result_HIV.txt")
countries <- lapply(medline_records, findCountry)
dates <- lapply(medline_records, findDate)

load("Workdata/dates.Rdata")
load("Workdata/countries.Rdata")

countries2 <- lapply(countries, function(x) {
  if(length(x) > 0) {
    len <- length(x)
    x[len]
  } else {
    NA
  }
```

```
})


world.cities <- read.csv("H:/Dokumenter/Forskning/Studier/Publication_by_country/R
awdata/WorldCities/worldcitiespop.txt",
                          stringsAsFactors = F)

no.count <- sapply(countries, function(x) identical(x, character(0)))
try.city <- medline_records[no.count]
all.new.countries <- lapply(try.city, findCountry3)

countries2[sapply(countries2, function(x) is.na(x))] <- all.new.countries
countries2 <- lapply(countries2, function(x) if(length(x) > 1) c(">1 country") els
e x)

df <- data.frame(cbind(do.call(rbind, countries2), do.call(rbind, dates)), strings
AsFactors = F) %>%
  mutate_each(funs(as.numeric), X2)
colnames(df) <- c("Country", "Year")

# all countries with HIV papers n = 194
unique.countries <- unique(unlist(countries2))
unique.countries[order(unique.countries)]

df1 <- mutate(df,
              region_origin = countrycode(Country, origin = "country.name", destin
ation = "region"))


# The code below looks busy in R markdown, but should be fine when used in R

df2 <- mutate(df1,
              region_origin2 = ifelse(region_origin == "Northern America",
                                      "Northern America",
                                      ifelse(region_origin %in% c("Western Europe"
,
                                                                  "Northern Europe
",
                                                                  "Southern Europe
"),
                                             "Western Europe",
                                             ifelse(region_origin == "Eastern Euro
pe",
                                                    "Eastern Europe",
                                                    ifelse(region_origin %in% c("S
outhern Africa",
                                                                                "M
iddle Africa",
                                                                                   "E
astern Africa",
                                                                                   "W
estern Africa"),
                                                            "Sub-Saharan Africa",
                                                            ifelse(region_origin ==
"Australia and New Zealand",
                                                                   "Australia and N
ew Zealand",
                                                                   ifelse(region_or
igin %in% c("South America",
"Caribbean",
"Central America"),
                                                                          "Latin/Ce
ntral America",
                                                                          ifelse(re
gion_origin %in% c("Southern Asia",
"Central Asia",
"Eastern Asia",
"South-Eastern Asia",
"Micronesia",
"Melanesia", "Polynesia"),
                                                                                 "A
sia and Oceania",
                                                                                 if
else(region_origin %in% c("Northern Africa",
"Western Asia"),
"North Africa and Middle East",
NA)))))))))) %>%
  group_by(region_origin2, Year) %>%
  summarise(total = n()) %>%
  mutate(total = cumsum(total)) %>%
  ungroup %>%
  complete(region_origin2, Year, fill = list(total = 0)) %>%
  mutate_each(funs(as.factor), region_origin2)

fig1 <- ggplot(df2, aes(x = Year, y = total, fill = region_origin2)) +
  geom_area() +
  xlim(1980,2018) +
  ylab("Cumulative number of HIV papers") +
  theme_bw() +
  theme(legend.title=element_blank()) +
  theme(legend.justification = c(-0.1,1), legend.position = c(0,0.97)) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank()) +
  theme(axis.text.y = element_text(angle = 90, hjust = 1)) +
  scale_fill_brewer(palette="YlOrRd") +
  scale_x_continuous(breaks = seq(1980, 2020, by = 5))

df3 <- group_by(df, Country) %>%
  summarize(total = n()) %>%
  mutate(no.f = cut(total,
                    breaks = c(0,10,100,1000,10000,100000),
                    labels = c("<10", "10-99","100-999","1000-9999",">10000")))


n <- joinCountryData2Map(df3,
                         joinCode="NAME",
                         nameJoinColumn="Country")

mapParams <- mapCountryData(n,
                            nameColumnToPlot="no.f",
                            mapTitle="",
                            catMethod="categorical",
                            addLegend = F)
do.call(addMapLegendBoxes,
        c(mapParams,
          x='bottom',
          horiz=TRUE,
          title="Cumulative number of papers",
          box.col = "white",
          cex=0.9))
```