R version 3.5.1 (2018-07-02) -- "Feather Spray"

Copyright (C) 2018 The R Foundation for Statistical Computing

Platform: x86\_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.

You are welcome to redistribute it under certain conditions.

Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.

Type 'contributors()' for more information and

'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or

'help.start()' for an HTML browser interface to help.

Type 'q()' to quit R.

> # title: lifetable.make-ccb.r

> # purpose: life tables for mssa/county/state

> # note: current plan is: state 1-yr, county 3-year, MSSA 5-year.

> # criteria: publish if it has at least NNN PY of exposure history + >=95% deaths geocoded.

>

> ## 1 SETUP ----------------------------------------------------------------------

>

> ## 1.1 packages

> .pkg <- c("data.table","tidyr","readr","readxl")

> .inst <- .pkg %in% installed.packages()

> if(length(.pkg[!.inst]) > 0) install.packages(.pkg[!.inst])

> lapply(.pkg, library, character.only=TRUE)

data.table 1.11.6 Latest news: r-datatable.com

[[1]]

[1] "data.table" "stats" "graphics" "grDevices" "utils" "datasets" "methods" "base"

[[2]]

[1] "tidyr" "data.table" "stats" "graphics" "grDevices" "utils" "datasets" "methods" "base"

[[3]]

[1] "readr" "tidyr" "data.table" "stats" "graphics" "grDevices" "utils" "datasets" "methods" "base"

[[4]]

[1] "readxl" "readr" "tidyr" "data.table" "stats" "graphics" "grDevices" "utils" "datasets" "methods" "base"

>

> ## 1.2 dataset flag

> whichData <- "real" # "real" or "fake" death data

>

> ###MCS###

> .securePath <- "g:/0.Secure.Data/myData/"

> #########

>

>

> ## 1.3 paths

> myDrive <- getwd()

> myPlace <- paste0(myDrive,"/myCBD")

> upPlace <- paste0(myDrive,"/myUpstream")

> LTplace <- paste0(upPlace,"/lifeTables/dataOut/")

>

> ## 1.4 links

> .cbdlink <- paste0(myPlace,"/myInfo/Tract to Community Linkage.csv") # map tract level GEOID to comID

> .countylink <- paste0(myPlace,"/myInfo/County Codes to County Names Linkage.xlsx") # map county names to codes

> .nxtract <- paste0(upPlace,"/lifeTables/dataOut/nxTract.rds") # input deaths by tract

> .nxmssa <- paste0(upPlace,"/lifeTables/dataOut/nxMSSA.rds") # input deaths by mssa

> .nxcounty <- paste0(upPlace,"/lifeTables/dataOut/nxCounty.rds") # input deaths by county

> .nxstate <- paste0(upPlace,"/lifeTables/dataOut/nxState.rds") # input deaths by state

> .dxtract <- paste0(upPlace,"/lifeTables/dataOut/dxTract.rds") # input deaths by tract

> .dxmssa <- paste0(upPlace,"/lifeTables/dataOut/dxMSSA.rds") # input deaths by mssa

> .dxcounty <- paste0(upPlace,"/lifeTables/dataOut/dxCounty.rds") # input deaths by county

> .dxstate <- paste0(upPlace,"/lifeTables/dataOut/dxState.rds") # input deaths by state

> #.dxtract <- paste0(upPlace,"/lifeTables/dataOut/\_MSreal\_dxTract.rds") # input deaths by tract

> #.dxmssa <- paste0(upPlace,"/lifeTables/dataOut/\_MSreal\_dxMSSA.rds") # input deaths by mssa

> #.dxcounty <- paste0(upPlace,"/lifeTables/dataOut/\_MSreal\_dxCounty.rds") # input deaths by county

> #.dxstate <- paste0(upPlace,"/lifeTables/dataOut/\_MSreal\_dxState.rds") # input deaths by state

>

>

> ###MCS###

> if (whichData == "real"){

+ .dxtract <- paste0(.securePath,"dxTract.rds") # output deaths by tract

+ .dxmssa <- paste0(.securePath,"dxMSSA.rds") # output deaths by mssa

+ .dxcounty <- paste0(.securePath,"dxCounty.rds") # output deaths by county

+ .dxstate <- paste0(.securePath,"dxState.rds") # output deaths by state

+ }

> #########

>

>

>

> ## 1.5 setwd

> setwd(myDrive)

>

> ## 2 DATASETS ----------------------------------------------------------------------

>

> ## 2.1 load datasets

> .dxtract <- readRDS(.dxtract)

> .dxmssa <- readRDS(.dxmssa)

> .dxcounty <- readRDS(.dxcounty)

> .dxstate <- readRDS(.dxstate)

> .nxtract <- readRDS(.nxtract)

> .nxmssa <- readRDS(.nxmssa)

> .nxcounty <- readRDS(.nxcounty)

> .nxstate <- readRDS(.nxstate)

>

>

>

>

>

>

>

>

> ## 3 DATA CLEANING ------------------------------------------------------------------

>

> ## 3.1 calculate what share of the deaths in the tract's parent county are not geocoded to the tract level

> cbd.link <- setDT(read\_csv(.cbdlink, col\_types="icccc")) # tract-to-MSSA-to-county crosswalk

> cbd.link[, countyFIPS:=substr(GEOID,1,5)]

>

> .dxcounty[,countyFIPS:=substring(GEOID,1,5)] # create trimmed countyFIPS in dxcounty

> .csum <- .dxcounty[year %in% 2015:2017,.(dx=sum(dx)),by=countyFIPS] # county level sum

> .tsum <- .dxtract[year %in% 2015:2017][

+ cbd.link,nomatch=0,on='GEOID'][

+ ,.(dx=sum(dx)),by=c("countyFIPS")] # county level sum from tracts

> .diff <- .csum[.tsum,on=c("countyFIPS")][,dpc:=(dx-i.dx)/dx] # mismatch between tracts-county total

> # may be negative

>

> ## 3.2 generate a list of unacceptably high missing tract level geocodes per county

> .missTract<-.diff[abs(dpc)>=.05][cbd.link,nomatch=0,on='countyFIPS'][,c("GEOID","dpc")]

> .missMSSA <-.diff[abs(dpc)>=.05][cbd.link,nomatch=0,on='countyFIPS'][,.(dpc=mean(dpc)),by="comID"]

>

> ## 3.1 temp rename comID to GEOID for consistency with functions using GEOID

> .dxmssa[,GEOID:=comID]

> .nxmssa[,GEOID:=comID]

> .missMSSA[,GEOID:=comID]

>

> ## 3.2 for fake data, weight exposure to accord to dx (censors geographies, but should provide more accurate ex)

> if (whichData=="fake") {

+ .actual<-data.table(year=c(2015:2018),dx=c(233143,239006,242461,248118,245349,258694,261712,267556,268661)) # actual

+ .sampled<-.dxstate[year>=2015 & year<=2017 & sex=="TOTAL",(dx=sum(dx))] # sampled deaths statewide 2013-2017

+ .factor<-.sampled/.actual[year %in% 2015:2017,.(sum(dx))][[1]] # ratio of sampled to actual deaths

+ .nxtract[,nx:=round(nx\*.factor)] # inflate deaths to compensate for sample size

+ .nxmssa[,nx:=round(nx\*.factor)] # inflate deaths to compensate for sample size

+ .nxcounty[,nx:=round(nx\*.factor)] # inflate deaths to compensate for sample size

+ .nxstate[,nx:=round(nx\*.factor)] # inflate deaths to compensate for sample size

+ }

>

> ## 3.3 function: calculate how many years of exposure data required for stable LT

> doCheckPY <- function(d=NULL,t=10000) { # d=population dataset, t=critical PY exposure

+ return(d[,.(nx=sum(nx)),by=.(GEOID,year,sex)][ # sum nx by geo/sex

+ nx<t,.(GEOID,sex, # filter out those with <minimum PY,

+ flag=ceiling((t/nx)/2))][ # calculate N years needed, rounded, by sex

+ ,.(flag=max(flag)),by=GEOID][flag>1]) # report if need more than 1 year

+ }

>

> ## 3.5 list of areas that will not be processed due to too few PY (to plug in to map as blanks)

> .zeroTract <-doCheckPY(.nxtract,t=10000)[flag>5] # tracts that require>N years data. (5/10/15k PY: 281/874/2543 of 9170)

> .zeroMSSA <-doCheckPY(.nxmssa,t=10000)[flag>5] # MSSAs that require>5 years data. (5/10/15k PY: 71/127/146 of 542)

> .zeroCounty<-doCheckPY(.nxcounty,t=10000)[flag>3] # counties that require>N years data. (5/10/15k PY: 5/11/17 of 58)

>

> ## 3.6 add to list of areas to skip due to too few geocoded deaths

> .zeroTract <- merge(.zeroTract,.missTract,all=TRUE,by="GEOID") # tract: 1264 censored

> .zeroMSSA <- merge(.zeroMSSA,.missMSSA,all=TRUE,by="GEOID") # MSSA: 165 censored

> .zeroMSSA[,comID:=GEOID]

>

> ## 3.7 function to generate an extract of years by geo

> ## tbd: pass varname as argument to function (dx or nx)

> doExtract <- function(d=NULL, nyrs=NA, end=2017, level=NA) {

+ start=end-nyrs

+ if (level=="tract") .zero<-.zeroTract

+ if (level=="county") .zero<-.zeroCounty

+ if (level=="mssa") {

+ d[,GEOID:=comID]

+ .zero<-.zeroMSSA

+ }

+ if (level=="state") .zero<-data.table(GEOID="")

+ tmp<-d[year>=start & year<=end & (GEOID %in% .zero$GEOID)==FALSE][,

+ .(dx=sum(dx)), by=c("GEOID","sex","agell","ageul")]

+ tmp[, year:=end]

+ tmp[, nyrs:=nyrs]

+ if (level=="mssa") {

+ tmp[,comID:=GEOID]

+ tmp[,GEOID:=NULL]

+ d[,GEOID:=NULL]

+ }

+ return(tmp)

+ }

>

> ## 3.8 summarize deaths

> # generate deaths time series, where each year represents a moving 5-year window of pooled deaths, e.g. 2015==2016-10

> # the code runs these two steps (step 1 is show for 1 year and then in an lapply)

> #ltdx.tract <- doExtractDx(d=.dxtract,nyrs=5,end=2017,level="tract")

> #ltdx.tract <- lapply(2015:2017,doExtractDx,d=.dxtract,nyrs=5,level="tract")

> #ltdx.tract <- data.table(do.call(rbind,ltdx.tract))

> ltdx.tract<-data.table(do.call(rbind,lapply(2015:2017,doExtract,d=.dxtract,nyrs=5,level="tract")))

> ltdx.mssa<-data.table(do.call(rbind,lapply(2015:2017,doExtract,d=.dxmssa,nyrs=5,level="mssa")))

> ltdx.county<-data.table(do.call(rbind,lapply(2015:2017,doExtract,d=.dxcounty,nyrs=3,level="county")))

> ltdx.state<-data.table(do.call(rbind,lapply(2015:2017,doExtract,d=.dxstate,nyrs=1,level="state")))

>

> ## 3.7 summarize exposures (rename nx as dx temporarily to use the above subroutine doExtract)

> .nxtract[,dx:=nx];

> ltnx.tract<-data.table(do.call(rbind,lapply(2015:2017,doExtract,d=.nxtract,nyrs=5,level="tract")))

> ltnx.tract[,nx:=dx]; ltnx.tract[,dx:=NULL]; .nxtract[,dx:=NULL]

> .nxmssa[,dx:=nx];

> ltnx.mssa<-data.table(do.call(rbind,lapply(2015:2017,doExtract,d=.nxmssa,nyrs=5,level="mssa")))

> ltnx.mssa[,nx:=dx]; ltnx.mssa[,dx:=NULL]; .nxmssa[,dx:=NULL]

> .nxcounty[,dx:=nx];

> ltnx.county<-data.table(do.call(rbind,lapply(2015:2017,doExtract,d=.nxcounty,nyrs=3,level="county")))

> ltnx.county[,nx:=dx]; ltnx.county[,dx:=NULL]; .nxcounty[,dx:=NULL]

> .nxstate[,dx:=nx];

> ltnx.state<-data.table(do.call(rbind,lapply(2015:2017,doExtract,d=.nxstate,nyrs=1,level="state")))

> ltnx.state[,nx:=dx]; ltnx.state[,dx:=NULL]; .nxstate[,dx:=NULL]

>

> ## 3.8 merge deaths and exposures

> ## tract

> setkeyv(ltnx.tract,c("GEOID","year","sex","agell","ageul"))

> setkeyv(ltdx.tract,c("GEOID","year","sex","agell","ageul"))

> mx.tract <-merge(ltnx.tract, ltdx.tract,

+ by=c("GEOID","year","sex","agell","ageul"), all=TRUE) # merge pop, death data

Warning message:

In head(key(i), length(leftcols)) == names(i)[leftcols] :

longer object length is not a multiple of shorter object length

> ## mssa

> setkeyv(ltnx.mssa,c("comID","year","sex","agell","ageul"))

> setkeyv(ltdx.mssa,c("comID","year","sex","agell","ageul"))

> mx.mssa <-merge(ltnx.mssa, ltdx.mssa,

+ by=c("comID","year","sex","agell","ageul"), all=TRUE) # merge pop, death data

Warning message:

In head(key(i), length(leftcols)) == names(i)[leftcols] :

longer object length is not a multiple of shorter object length

> ## county

> setkeyv(ltnx.county,c("GEOID","year","sex","agell","ageul"))

> setkeyv(ltdx.county,c("GEOID","year","sex","agell","ageul"))

> mx.county <-merge(ltnx.county, ltdx.county,

+ by=c("GEOID","year","sex","agell","ageul"), all=TRUE) # merge pop, death data

> mx.county[sex=="TOTAL" & year==2017 & GEOID=="06037000000"][

+ ,.(nx=sum(nx),dx=sum(dx))] # check county PY and Dx (3 yrs combined)

nx dx

1: 40827931 245605

> ## state

> setkeyv(ltnx.state,c("GEOID","year","sex","agell","ageul"))

> setkeyv(ltdx.state,c("GEOID","year","sex","agell","ageul"))

> mx.state <-merge(ltnx.state, ltdx.state,

+ by=c("GEOID","year","sex","agell","ageul"), all=TRUE) # merge pop, death data

> mx.state[sex=="TOTAL" & year==2017][,.(nx=sum(nx),dx=sum(dx))] # check state PY and Dx (1 year)

nx dx

1: 79048059 524820

>

> ## 3.9 rectangularize and collapse by new age groups

> ## i=id for each life table. ageul missing after 'complete' step

> ## tract

> mx.tract<-setDT(complete(mx.tract,GEOID,sex,agell)) # (tidyr) rectangularize and key as DT

> mx.tract[is.na(nx), nx:=0] # fill in new missing values with 0

> mx.tract[is.na(dx), dx:=0]

> mx.tract[, i:=.GRP, by=c("GEOID","sex","year")] # create an ID variable for each LT

> setkeyv(mx.tract,c("i","agell"))

> ## mssa

> mx.mssa<-setDT(complete(mx.mssa,comID,sex,agell)) # (tidyr) rectangularize and key as DT

> mx.mssa[is.na(nx), nx:=0] # fill in new missing values with 0

> mx.mssa[is.na(dx), dx:=0]

> mx.mssa[, i:=.GRP, by=c("comID","sex","year")] # create an ID variable for each LT

> setkeyv(mx.mssa,c("i","agell"))

> ## county

> mx.county[is.na(nx), nx:=0] # fill in new missing values with 0

> mx.county[is.na(dx), dx:=0]

> mx.county[, i:=.GRP, by=c("GEOID","sex","year")] # create an ID variable for each LT

> setkeyv(mx.county,c("i","agell"))

> ## state

> mx.state[is.na(nx), nx:=0] # fill in new missing values with 0

> mx.state[is.na(dx), dx:=0]

> mx.state[, i:=.GRP, by=c("GEOID","sex","year")] # create an ID variable for each LT

> setkeyv(mx.state,c("i","agell"))

>

> ## 3.1 figure for CONSORT style flowchart

> ## tract

> if (whichData=="real") {

+ length(unique(.nxtract[,GEOID])) # n total tracts: 9170

+ length(unique(.nxtract[year %in% 2013:2017, .(nx=sum(nx)), by=GEOID][!is.na(nx),GEOID]))

+ # n tracts with data for 2013-17: 8057

+ length(unique(.nxtract[year %in% 2013:2017, .(nx=sum(nx)), by=GEOID][nx==0,GEOID]))

+ # n tracts with 0 population: 38 (excluded)

+ length(unique(.nxtract[year %in% 2013:2017, .(nx=sum(nx)), by=GEOID][nx>=1 & nx<10000,GEOID]))

+ # n tracts w/1-9999 exposure in preceding 5 yrs: 84

+ length(unique(.nxtract[year %in% 2013:2017, .(nx=sum(nx)), by=GEOID][nx>=10000,GEOID]))

+ # n tracts w/10k+ exposure in preceding 5 yrs: 7936

+ length(unique(mx.tract[sex=="TOTAL",GEOID])) # n valid tracts with 10k+95% geocoded: 6907

+ length(unique(mx.tract[sex=="TOTAL" & dx>=nx,GEOID])) # n tracts w/more deaths than nx estimated (IMPUTE LATER): ???

+ length(mx.tract[sex=="TOTAL" & dx==0,.(n=length(dx)),by=GEOID][n>=1 & n<=4,GEOID]) # n tracts w/1-4 or empty death cells (IMPUTE LATER): ???

+ length(mx.tract[sex=="TOTAL" & dx==0,.(n=length(dx)),by=GEOID][n>=5,GEOID]) # n tracts w/5+ empty death cells (IMPUTE LATER): ???

+ }

[1] 6426

>

> ## 4 ANALYSIS (LIFE TABLE) ----------------------------------------------------------

> ## - there are 9 years of exposure (population) data, so that is a limit of ACS.

> ## - rules of thumb are 10,000 or 15k PY of exposure for a stable LT in a high-e0 population.

> ## - earlier versions return a table of start/end years needed, working back from 2017.

> ## - temporarily, we are using a 5 years window for all tracts, 3 years for counties, 1 year for state level life table.

> ## - ACS population estimates include uncertainty, not accounted for here.

> ## - for this version, no spatiotemporal smoothing.

> ## - for time series, moving window, e.g. 2015 is from pooled 2006-10 ACS, 2011 from pooled 2007-11, etc.

>

> ## 4.1 generic function to produce a life table from minimum inputs

> ## x is a vector of age groups, nx is the corresponding vector of pop, dx of deaths

> ## sex is M or MALE or F or FEMALE (used to calc ax); ax is an optional vector of ax values

> doLT <- function(x, Nx, Dx, sex, ax=NULL) {

+ m <- length(x) # get final age group by length of age vector

+ mx <- Dx/Nx # mortality rate

+ n <- c(diff(x), NA) # n years between age groups

+ if(is.null(ax)){

+ ax <- rep(0,m)

+ if(x[1]!=0 | x[2]!=1){

+ ax <- n/2 # rule of thumb: 1/2 age interval

+ ax[m] <- 1 / mx[m] # rule of thumb: inverse of mx in final age interval

+ }

+ else{

+ if(grepl("F",sex[1])){ # ax values for women from Coale (Preston et al 2001)

+ if(mx[1]>=0.107) {

+ ax[1] <- 0.350

+ }

+ else{

+ ax[1] <- 0.053 + 2.800\*mx[1]

+ }

+ }

+ if(!grepl("F",sex[1])){ # ax values for men

+ if(mx[1]>=0.107) {

+ ax[1] <- 0.330

+ }

+ else{

+ ax[1] <- 0.045 + 2.684\*mx[1]

+ }

+ }

+ ax[-1] <- n[-1]/2

+ ax[m] <- 1 / mx[m]

+ }

+ }

+ qx <- n\*mx / (1 + (n - ax) \* mx) # probablity of death (from mortality rate)

+ qx[m] <- 1 # 100% at oldest age group

+ px <- 1-qx # pr(survival)

+ lx <- cumprod(c(1,px))\*100000 # 100,000 for radix

+ dx <- -diff(lx) # deaths each age interval

+ Lx <- n\*lx[-1] + ax\*dx # PY lived in this age group

+ lx <- lx[-(m+1)] # survivors

+ Lx[m] <- lx[m]/mx[m] # PY lived in final age group

+ Lx[is.na(Lx)|is.infinite(Lx)] <- 0 # in case of NA or Inf values from poorly formed LTs

+ Tx <- rev(cumsum(rev(Lx))) # cumulative PY lived at this age and above

+ ex <- Tx/lx # life expectancy at this age

+ return(data.table(x, n, Nx, Dx, mx, ax, qx, px, lx, dx, Lx, Tx, ex))

+ }

>

> ## 4.2 call to LT function

> ## tract

> lt.tract<-data.table() # init empty dt

> .pb <- txtProgressBar(min = 0, max = mx.tract[agell==0,.N], style = 3) # show a text progress bar for loop

| | 0%> for (j in 1:mx.tract[agell==0,.N]) {

+ x<-mx.tract[i==j,agell]

+ nx<-mx.tract[i==j,nx]

+ dx<-mx.tract[i==j,dx]

+ sex<-mx.tract[i==j,sex]

+ lt.tract <- rbindlist(list(lt.tract, # fast rbind results

+ cbind(i=mx.tract[i==j,i], # attach ID to life table

+ GEOID=mx.tract[i==j,GEOID],

+ sex,

+ year=mx.tract[i==j,year],

+ doLT(x,nx,dx,sex))))

+ setTxtProgressBar(.pb,j)

+ }

|==============================================================================================================================================| 100%> close(.pb)

> setkeyv(lt.tract,c("i","x"))

> ## mssa

> lt.mssa<-data.table() # init empty dt

> .pb <- txtProgressBar(min = 0, max = mx.mssa[agell==0,.N], style = 3) # show a text progress bar for loop

| | 0%> for (j in 1:mx.mssa[agell==0,.N]) {

+ x<-mx.mssa[i==j,agell]

+ nx<-mx.mssa[i==j,nx]

+ dx<-mx.mssa[i==j,dx]

+ sex<-mx.mssa[i==j,sex]

+ lt.mssa <- rbindlist(list(lt.mssa, # fast rbind result to lt.county

+ cbind(i=mx.mssa[i==j,i], # attach ID to life table

+ comID=mx.mssa[i==j,comID],

+ sex,

+ year=mx.mssa[i==j,year],

+ doLT(x,nx,dx,sex))))

+ setTxtProgressBar(.pb,j)

+ }

|==============================================================================================================================================| 100%> close(.pb)

> setkeyv(lt.mssa,c("i","x"))

> ## county

> lt.county<-data.table() # init empty dt

> .pb <- txtProgressBar(min = 0, max = mx.county[agell==0,.N], style = 3) # show a text progress bar for loop

| | 0%> for (j in 1:mx.county[agell==0,.N]) {

+ x<-mx.county[i==j,agell]

+ nx<-mx.county[i==j,nx]

+ dx<-mx.county[i==j,dx]

+ sex<-mx.county[i==j,sex]

+ lt.county <- rbindlist(list(lt.county, # fast rbind result to lt.county

+ cbind(i=mx.county[i==j,i], # attach ID to life table

+ GEOID=mx.county[i==j,GEOID],

+ sex,

+ year=mx.county[i==j,year],

+ doLT(x,nx,dx,sex))))

+ setTxtProgressBar(.pb,j)

+ }

|==============================================================================================================================================| 100%> close(.pb)

> setkeyv(lt.county,c("i","x"))

> lt.county[GEOID=="06001000000" & sex=="TOTAL" & x==0, c("GEOID","sex","year","ex")] # ALAMEDA COUNTY (CHECK)

GEOID sex year ex

1: 06001000000 TOTAL 2015 82.31450

2: 06001000000 TOTAL 2016 82.52477

3: 06001000000 TOTAL 2017 82.78867

> lt.county[GEOID=="06033000000" & sex=="TOTAL" & x==0, c("GEOID","sex","year","ex")] # LAKE COUNTY (CHECK)

GEOID sex year ex

1: 06033000000 TOTAL 2015 75.10905

2: 06033000000 TOTAL 2016 75.16956

3: 06033000000 TOTAL 2017 75.20999

> lt.county[GEOID=="06115000000" & sex=="TOTAL" & x==0, c("GEOID","sex","year","ex")] # YUBA COUNTY (CHECK)

GEOID sex year ex

1: 06115000000 TOTAL 2015 76.74350

2: 06115000000 TOTAL 2016 76.41595

3: 06115000000 TOTAL 2017 75.93547

> ## state

> lt.state<-data.table() # init empty dt

> for (j in 1:mx.state[agell==0,.N]) {

+ x<-mx.state[i==j,agell]

+ nx<-mx.state[i==j,nx]

+ dx<-mx.state[i==j,dx]

+ sex<-mx.state[i==j,sex]

+ lt.state <- rbindlist(list(lt.state, # fast rbind result to lt.state

+ cbind(i=mx.state[i==j,i], # attach ID to life table

+ GEOID=mx.state[i==j,GEOID],

+ sex,

+ year=mx.state[i==j,year],

+ doLT(x,nx,dx,sex))))

+ }

> setkeyv(lt.state,c("i","x"))

> lt.state[x==0 & sex=="TOTAL",c("GEOID","sex","year","ex")] # CA state e0

GEOID sex year ex

1: 06000000000 TOTAL 2015 81.76716

2: 06000000000 TOTAL 2016 81.73091

3: 06000000000 TOTAL 2017 81.88739

>

> ## 4.3 function to produce a life table from qx values only (used in simulation for CI)

> doQxLT<- function(x, qx, sex, ax=NULL, last.ax=5) {

+ m <- length(x)

+ n <- c(diff(x), NA)

+ qx[is.na(qx)] <- 0

+ if(is.null(ax)){

+ ax <- rep(0,m)

+ if(x[1]!=0 | x[2]!=1){

+ ax <- n/2

+ ax[m] <- last.ax

+ }

+ else{

+ if(grepl("F",sex[1])){

+ if(qx[1]>=0.1){

+ ax[1] <- 0.350

+ }

+ else{

+ ax[1] <- 0.05 + 3\*qx[1]

+ }

+ }

+ if(!grepl("F",sex[1])){

+ if(qx[1]>=0.1){

+ ax[1] <- 0.33

+ }

+ else{

+ ax[1] <- 0.0425 + 2.875\*qx[1]

+ }

+ }

+ ax[-1] <- n[-1]/2

+ ax[m] <- last.ax

+ }

+ }

+ px <- 1-qx

+ lx <- cumprod(c(1,px))\*100000

+ dx <- -diff(lx)

+ Lx <- n\*lx[-1] + ax\*dx

+ lx <- lx[-(m+1)]

+ Lx[m] <- lx[m]\*last.ax

+ Lx[is.na(Lx)|is.infinite(Lx)] <- 0 # fix NA or Inf values

+ Tx <- rev(cumsum(rev(Lx)))

+ ex <- Tx/lx

+ return(data.table(x, n, ax, qx, px, lx, dx, Lx, Tx, ex))

+ }

>

> ## 4.4 function to calculate confidence interval around life expectancy

> ## - E. Andreev and V. Shkolnikov. 2015. "Spreadsheet for calculation of confidence limits

> ## for any life table or healthy-life table quantity". MPIDR Technical Report 2015-005.

> doLTCI <- function(LT=NULL, # LT matrix created by doLT

+ which.x=0, # CI of ex at which age?

+ ns=1000, # N simulations

+ level=0.95) { # desired CI

+ setDT(LT) # (redundant if already DT)

+ m <- LT[,.N] # N age groups == n rows

+ x <- LT[,x] # ages

+ qx <- LT[,qx] # qx

+ Dx <- LT[,Dx] # Dx

+ .trials <- round(Dx/qx) # trials for binomial, rounded

+ .lastax <- LT$ax[m] # ax in open-ended age group

+ Y <- suppressWarnings(matrix(rbinom(m\*ns,.trials,qx), # simulated death counts (binomial)

+ m,ns))

+ QX <- Y/.trials # simulated qx

+ wh <- which(x==which.x) # row number which contains age group of interest

+ fun.ex <- function(qx) { # subroutine to compute ex for simulated qx

+ return(doQxLT(x=x, qx, sex=sex, last.ax=.lastax)$ex[wh]) # runs doQxLT and saves one value of ex

+ }

+ exsim <- apply(QX, 2, fun.ex) # call subroutine

+ CI <- quantile(exsim, # generate CI

+ probs = c((1-level)/2,

+ 1 - (1-level)/2))

+ return(list(ex=LT$ex[wh], # result is list of: ex= life table ex

+ meanex=mean(exsim), # meanex: mean of simulated ex

+ ciex=CI, # ciex: ci around ex

+ exsim=exsim, # every simulated ex result

+ which.x=which.x)) # the age

+ }

>

> ## 4.5 call to CI function

> ##

> ## tract

> ltci.tract<-data.table() # initialize an empty DT

> .counter<-lt.tract[x==0,.N]

> .pb <- txtProgressBar(min = 0, max = .counter, style = 3) # show a text progress bar for loop

| | 0%> for (j in 1:lt.tract[x==0,.N]) { # or "for (j in 1:2) {" for a quick test

+ ltci.tract<-rbindlist(list(ltci.tract, # fast rbind result to ltci.tract (2 items)

+ cbind(data.table( # format results of exsim as DT

+ t(unlist(doLTCI(lt.tract[i==j], # run specific LT

+ which.x=0,ns=500,level=.95)[ # pass parameters for simulation

+ c(1,2,3,5)]))) # save just desired fields from exsim

+ ,j) # attach ID to simulation results;

+ )

+ )

+ setTxtProgressBar(.pb,j)

+ }

|============================================================= | 43%Error in rbindlist(list(ltci.tract, cbind(data.table(t(unlist(doLTCI(lt.tract[i == :

Item 2 has 5 columns, inconsistent with item 1 which has 6 columns. If instead you need to fill missing columns, use set argument 'fill' to TRUE.

> names(ltci.tract)[names(ltci.tract) == "j"] = "i" # rename j column to i (ID of mx file)

> names(ltci.tract)[names(ltci.tract) == "which.x"] = "agell" # rename to agell

> setkeyv(ltci.tract,c("i","agell"))

> ltci.tract<-ltci.tract[mx.tract[agell==0,c("i","agell","sex","GEOID","year")],nomatch=0] # merge sex and geo identifiers

> ##

> ## mssa

> ltci.mssa<-data.table() # initialize an empty DT

> .counter<-lt.mssa[x==0,.N]

> .pb <- txtProgressBar(min = 0, max = .counter, style = 3) # show a text progress bar for loop

| | 0%> for (j in 1:lt.mssa[x==0,.N]) { # or "for (j in 1:2) {" for a quick test

+ ltci.mssa<-rbindlist(list(ltci.mssa, # fast rbind result to ltci.mssa (2 items)

+ cbind(data.table( # format results of exsim as DT

+ t(unlist(doLTCI(lt.mssa[i==j], # run specific LT

+ which.x=0,ns=500,level=.95)[ # pass parameters for simulation

+ c(1,2,3,5)]))) # save just desired fields from exsim

+ ,j) # attach ID to simulation results;

+ )

+ )

+ setTxtProgressBar(.pb,j)

+ }

|==============================================================================================================================================| 100%> close(.pb)

> names(ltci.mssa)[names(ltci.mssa) == "j"] = "i" # rename j column to i (ID of mx file)

> names(ltci.mssa)[names(ltci.mssa) == "which.x"] = "agell" # rename to agell

> setkeyv(ltci.mssa,c("i","agell"))

> ltci.mssa<-ltci.mssa[mx.mssa[agell==0,c("i","agell","sex","comID","year")],nomatch=0] # merge sex and geo identifiers

> ##

> ## county

> ltci.county<-data.table() # initialize an empty DT

> .counter<-lt.county[x==0,.N]

> .pb <- txtProgressBar(min = 0, max = .counter, style = 3) # show a text progress bar for loop

| | 0%> for (j in 1:lt.county[x==0,.N]) { # or "for (j in 1:2) {" for a quick test

+ ltci.county<-rbindlist(list(ltci.county, # fast rbind result to ltci.county (2 items)

+ cbind(data.table( # format results of exsim as DT

+ t(unlist(doLTCI(lt.county[i==j], # run specific LT

+ which.x=0,ns=500,level=.95)[ # pass parameters for simulation

+ c(1,2,3,5)]))) # save just desired fields from exsim

+ ,j) # attach ID to simulation results;

+ )

+ )

+ setTxtProgressBar(.pb,j)

+ }

|==============================================================================================================================================| 100%> names(ltci.county)[names(ltci.county) == "j"] = "i" # rename j column to i (ID of mx file)

> names(ltci.county)[names(ltci.county) == "which.x"] = "agell" # rename to agell

> setkeyv(ltci.county,c("i","agell"))

> ltci.county<-ltci.county[mx.county[agell==0,c("i","agell","sex","GEOID","year")],nomatch=0] # merge sex and geo identifiers

> ##

> ## state

> ltci.state<-data.table() # initialize an empty DT

> for (j in 1:lt.state[x==0,.N]) { # or "for (j in 1:2) {" for a quick test

+ ltci.state<-rbindlist(list(ltci.state, # fast rbind result to ltci.state (2 items)

+ cbind(data.table( # format results of exsim as DT

+ t(unlist(doLTCI(lt.state[i==j], # run specific LT

+ which.x=0,ns=500,level=.95)[ # pass parameters for simulation

+ c(1,2,3,5)]))) # save just desired fields from exsim

+ ,j) # attach ID to simulation results;

+ )

+ )

+ }

> names(ltci.state)[names(ltci.state) == "j"] = "i" # rename j column to i (ID of mx file)

> names(ltci.state)[names(ltci.state) == "which.x"] = "agell" # rename to agell

> setkeyv(ltci.state,c("i","agell"))

> ltci.state<-ltci.state[mx.state[x==0,c("i","agell","sex","GEOID","year")],nomatch=0] # merge sex and geo identifiers

Error: i evaluates to a logical vector length 19 but there are 171 rows. Recycling of logical i is no longer allowed as it hides more bugs than is worth the rare convenience. Explicitly use rep(...,length=.N) if you really need to recycle.

>

> ## 5 DIAGNOSTICS ----------------------------------------------------------

>

> ## 5.1 diagnostic plots

> ## - these are of a de novo simulation, not the output dataset

> ## - use to check any single LT for typical results, by its idx value

> doExHist <- function(dat=NULL,idx=NULL,age=0,reps=500,ci=.95) {

+ while (!is.null(dev.list())) dev.off()

+ par(mfrow=c(1,1))

+ tmp<-doLTCI(LT=dat[i==idx],age,reps,ci)

+ hist(tmp$exsim)

+ abline(v=tmp$ex,col=2)

+ abline(v=tmp$meanex,col=4)

+ abline(v=tmp$ciex,col=4)

+ }

> doExHist(dat=lt.tract,idx=1,age=0,reps=500,ci=.9)

> doExHist(dat=lt.mssa,idx=1,age=0,reps=500,ci=.9)

> doExHist(dat=lt.county,idx=1,age=0,reps=500,ci=.9)

> doExHist(dat=lt.state,idx=1,age=0,reps=500,ci=.9)

>

> ## 6 EXPORT DATA ----------------------------------------------------------

>

> ## 6.1 export datasets

> saveRDS(ltci.tract, file=paste0(LTplace,"LTciTract.rds")) # GEOID sex (char) x (age0) ex meanex ciex.low ciex.high

> saveRDS(ltci.mssa, file=paste0(LTplace,"LTciMSSA.rds")) # comID sex (char) x (age0) ex meanex ciex.low ciex.high

> saveRDS(ltci.county, file=paste0(LTplace,"LTciCounty.rds")) # GEOID sex (char) x (age0) ex meanex ciex.low ciex.high

> saveRDS(ltci.state, file=paste0(LTplace,"LTciState.rds")) # GEOID sex (char) x (age0) ex meanex ciex.low ciex.high

>

> ## 6.3 compare with results from "real"

> ltci.tmp.c<-setDT(readRDS("C:/Users/fieshary/projects/CACommunityBurden/myUpstream/lifeTables/dataOut/\_MSreal\_LTciCounty.rds"))

Error in gzfile(file, "rb") : cannot open the connection

In addition: Warning message:

In gzfile(file, "rb") :

cannot open compressed file 'C:/Users/fieshary/projects/CACommunityBurden/myUpstream/lifeTables/dataOut/\_MSreal\_LTciCounty.rds', probable reason 'No such file or directory'

> ltci.tmp.m<-setDT(readRDS("C:/Users/fieshary/projects/CACommunityBurden/myUpstream/lifeTables/dataOut/\_MSreal\_LTciMSSAb.rds"))

Error in gzfile(file, "rb") : cannot open the connection

In addition: Warning message:

In gzfile(file, "rb") :

cannot open compressed file 'C:/Users/fieshary/projects/CACommunityBurden/myUpstream/lifeTables/dataOut/\_MSreal\_LTciMSSAb.rds', probable reason 'No such file or directory'

>

> # end