Package 'SUMMER'

May 7, 2026

Type Package

Title Small-Area-Estimation Unit/Area Models and Methods for Estimation in R

Version 2.0.0

Date 2025-01-03

Description Provides methods for spatial and spatio-temporal smoothing of demographic and health indicators using survey data, with particular focus on estimating and projecting under-five mortality rates, described in Mercer et al. (2015) <doi:10.1214/15-AOAS872>, Li et al. (2019) <doi:10.1371/journal.pone.0210645>, Wu et al. (DHS Spatial Analysis Reports No. 21, 2021), and Li et al. (2023) <doi:10.48550/arXiv.2007.05117>.

URL https://github.com/richardli/SUMMER, https://richardli.github.io/SUMMER/

BugReports https://github.com/richardli/SUMMER/issues

Depends R (>= 3.5)

License GPL (>= 2)

Imports survey, stats, spdep, survival, ggplot2, scales, utils, Matrix, reshape2, viridis, sp, sf, shadowtext, ggridges, methods, data.table, RColorBrewer, grDevices, fields, terra, haven, lifecycle

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Additional_repositories https://inla.r-inla-download.org/R/testing/

Suggests INLA, sn, knitr, rmarkdown, readstata13, patchwork, rdhs, R.rsp, sae, dplyr, tidyr, raster, fmesher, testthat (>= 3.0.0)

VignetteBuilder R.rsp, knitr

NeedsCompilation no

Config/build/clean-inst-doc FALSE

Config/testthat/edition 3

Config/testthat/parallel TRUE

Author Zehang R Li [cre, aut],
Bryan D Martin [aut],
Yuan Hsiao [aut],
Jessica Godwin [aut],
John Paige [aut],
Peter Gao [aut],
Jon Wakefield [aut],
Samuel J Clark [aut],
Geir-Arne Fuglstad [aut],
Andrea Riebler [aut]

Maintainer Zehang R Li <lizehang@gmail.com>

Repository CRAN

Date/Publication 2025-01-07 22:20:10 UTC

Contents

`aggPixelPreds` *Helper function of* `pixelPopToArea`

Description

Aggregates population from the pixel level to the level of the area of interest.

Usage

```
aggPixelPreds(
  Zg,
  Ng,
  areas,
  urban = target.pop.mat$urban,
  target.pop.mat = NULL,
  use.density = FALSE,
  stratify.by.urban = TRUE,
  normalize = use.density
)
```

Arguments

Zg	nIntegrationPoint x nsim matrix of simulated response (population numerators) for each pixel and sample
Ng	nIntegrationPoint x nsim matrix of simulated counts (population denominators) for each pixel and sample
areas	nIntegrationPoint length character vector of areas (or subareas)
urban	nIntegrationPoint length vector of indicators specifying whether or not pixels are urban or rural
target.pop.mat	same as in simPopCustom
use.density	whether to use population density as aggregation weights.
stratify.by.urban	whether or not to stratify simulations by urban/rural classification
normalize	if TRUE, pixel level aggregation weights within specified area are normalized to sum to 1. This produces an average of the values in Zg rather than a sum. In general, should only be set to TRUE for smooth integrals of risk.

---

aggPop *Aggregate populations to the specified areal level*

Description

Takes simulated populations and aggregates them to the specified areal level. Also calculates the aggregated risk and prevalence.

Usage

```
pixelPopToArea(
  pixel.level.pop,
  ea.samples,
  areas,
  stratify.by.urban = TRUE,
```

```
  target.pop.mat = NULL,
  do.fine.scale.risk = !is.null(pixel.level.pop$fineScaleRisk$p),
  do.smooth.risk = !is.null(pixel.level.pop$smoothRisk$p)
)

areaPopToArea(
  area.level.pop,
  areas.from,
  areas.to,
  stratify.by.urban = TRUE,
 do.fine.scale.risk = !is.null(area.level.pop$aggregationResults$pFineScaleRisk),
  do.smooth.risk = !is.null(area.level.pop$aggregationResults$pSmoothRisk)
)
```

Arguments

`pixel.level.pop`	pixel level population information that we want aggregate. In the same format as output from `simPopCustom`
`ea.samples`	nIntegrationPoint x nsim matrix of the number of enumeration areas per pixel sampled in the input pixel level population
`areas`	character vector of length nIntegrationPoints of area names over which we want to aggregate. Can also be subareas
`stratify.by.urban`	whether or not to stratify simulations by urban/rural classification
`target.pop.mat`	pixellated grid data frame with variables `lon`, `lat`, pop (target population), `area`, `subareas` (if subarea.level is TRUE), `urban` (if stratify.by.urban is TRUE), `east`, and `north`
`do.fine.scale.risk`	whether or not to calculate the fine scale risk in addition to the prevalence. See details
`do.smooth.risk`	Whether or not to calculate the smooth risk in addition to the prevalence. See details
`area.level.pop`	output of `simPopCustom` containing pixel level information about the population of interest
`areas.from`	character vector of length equal to the number of areas from which we would like to aggregate containing the unique names of the areas. Can also be subareas, but these are smaller than the "to areas", and each "from area" must be entirely contained in a single "to area"
`areas.to`	character vector of length equal to the number of areas from which we would like to aggregate containing the names of the areas containing with each respective 'from' area. Can also be a set of subareas, but these are larger than the "from areas".

Details

[Experimental]

Value

A list containing elements `fineScalePrevalence` and `fineScaleRisk`. Each of these are in turn lists with aggregated prevalence and risk for the area of interest, containg the following elements, were paranethesis indicate the elements for the fineScaleRisk model rather than fineScalePrevalence:

`p`	Aggregated prevalence (risk), calculated as aggregate of Z divided by aggregate of N
`Z`	Aggregated (expected) population numerator
`N`	Aggregated (expected) population denominator
`pUrban`	Aggregated prevalence (risk) in urban part of the area, calculated as aggregate of Z divided by aggregate of N
`ZUrban`	Aggregated (expected) population numerator in urban part of the area
`NUrban`	Aggregated (expected) population denominator in urban part of the area
`pRural`	Aggregated prevalence (risk) in rural part of the area, calculated as aggregate of Z divided by aggregate of N
`ZRural`	Aggregated (expected) population numerator in rural part of the area
`NRural`	Aggregated (expected) population denominator in rural part of the area
`A`	Aggregation matrix used to aggregate from pixel level to areal level
`AUrban`	Aggregation matrix used to aggregate from pixel level to urban part of the areal level
`ARural`	Aggregation matrix used to aggregate from pixel level to rural part of the areal level

Functions

- `pixelPopToArea()`: Aggregate from pixel to areal level
- `areaPopToArea()`: Aggregate areal populations to another areal level

Author(s)

John Paige

References

Paige, John, Geir-Arne Fuglstad, Andrea Riebler, and Jon Wakefield. "Spatial aggregation with respect to a population distribution: Impact on inference." Spatial Statistics 52 (2022): 100714.

See Also

`areaPopToArea`

Examples

```
## Not run:
# download Kenya GADM shapefiles from SUMMERdata github repository
githubURL <- paste0("https://github.com/paigejo/SUMMERdata/blob/main/data/",
                    "kenyaMaps.rda?raw=true")
tempDirectory = "~/"
mapsFilename = paste0(tempDirectory, "/kenyaMaps.rda")
if(!file.exists(mapsFilename)) {
  download.file(githubURL,mapsFilename)
}

# load it in
out = load(mapsFilename)
out
kenyaMesh <- fmesher::fm_as_fm(kenyaMesh)

adm1@data$NAME_1 = as.character(adm1@data$NAME_1)
adm1@data$NAME_1[adm1@data$NAME_1 == "Trans Nzoia"] = "Trans-Nzoia"
adm1@data$NAME_1[adm1@data$NAME_1 == "Elgeyo-Marakwet"] = "Elgeyo Marakwet"
adm2@data$NAME_1 = as.character(adm2@data$NAME_1)
adm2@data$NAME_1[adm2@data$NAME_1 == "Trans Nzoia"] = "Trans-Nzoia"
adm2@data$NAME_1[adm2@data$NAME_1 == "Elgeyo-Marakwet"] = "Elgeyo Marakwet"

# some Admin-2 areas have the same name
adm2@data$NAME_2 = as.character(adm2@data$NAME_2)
adm2@data$NAME_2[(adm2@data$NAME_1 == "Bungoma") &
                   (adm2@data$NAME_2 == "Lugari")] = "Lugari, Bungoma"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Kakamega") &
                   (adm2@data$NAME_2 == "Lugari")] = "Lugari, Kakamega"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Meru") &
                   (adm2@data$NAME_2 == "Igembe South")] = "Igembe South, Meru"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Tharaka-Nithi") &
                   (adm2@data$NAME_2 == "Igembe South")] = "Igembe South, Tharaka-Nithi"

# The spatial area of unknown 8 is so small, it causes problems unless its removed or
# unioned with another subarea. Union it with neighboring Kakeguria:
newadm2 = adm2
unknown8I = which(newadm2$NAME_2 == "unknown 8")
newadm2$NAME_2[newadm2$NAME_2 %in% c("unknown 8", "Kapenguria")] <-
  "Kapenguria + unknown 8"
admin2.IDs <- newadm2$NAME_2

newadm2@data = cbind(newadm2@data, NAME_2OLD = newadm2@data$NAME_2)
newadm2@data$NAME_2OLD = newadm2@data$NAME_2
newadm2@data$NAME_2 = admin2.IDs
newadm2$NAME_2 = admin2.IDs
temp <- terra::aggregate(as(newadm2, "SpatVector"), by="NAME_2")

library(sf)
temp <- sf::st_as_sf(temp)
temp <- sf::as_Spatial(temp)
```

```
tempData = newadm2@data[-unknown8I,]
tempData = tempData[order(tempData$NAME_2),]
newadm2 <- sp::SpatialPolygonsDataFrame(temp, tempData, match.ID = F)
adm2 = newadm2

# download 2014 Kenya population density TIF file

githubURL <- paste0("https://github.com/paigejo/SUMMERdata/blob/main/data/",
                    "Kenya2014Pop/worldpop_total_1y_2014_00_00.tif?raw=true")
popTIFFilename = paste0(tempDirectory, "/worldpop_total_1y_2014_00_00.tif")
if(!file.exists(popTIFFilename)) {
  download.file(githubURL,popTIFFilename)
}

# load it in
pop = terra::rast(popTIFFilename)

east.lim = c(-110.6405, 832.4544)
north.lim = c(-555.1739, 608.7130)

## Construct poppsubKenya, a table of urban/rural general population totals
## in each subarea. Technically, this is not necessary since we can load in
## poppsubKenya via data(kenyaPopulationData). First, we will need to calculate
## the areas in km^2 of the areas and subareas

# use Lambert equal area projection of areas (Admin-1) and subareas (Admin-2)
midLon = mean(adm1@bbox[1,])
midLat = mean(adm1@bbox[2,])
p4s = paste0("+proj=laea +x_0=0 +y_0=0 +lon_0=", midLon,
             " +lat_0=", midLat, " +units=km")

adm1_sf = st_as_sf(adm1)
adm1proj_sf = st_transform(adm1_sf, p4s)
adm1proj = as(adm1proj_sf, "Spatial")

adm2_sf = st_as_sf(adm2)
adm2proj_sf = st_transform(adm2_sf, p4s)
adm2proj = as(adm2proj_sf, "Spatial")

# now calculate spatial area in km^2
admin1Areas = as.numeric(st_area(adm1proj_sf))
admin2Areas = as.numeric(st_area(adm2proj_sf))

areapaKenya = data.frame(area=adm1proj@data$NAME_1, spatialArea=admin1Areas)
areapsubKenya = data.frame(area=adm2proj@data$NAME_1, subarea=adm2proj@data$NAME_2,
                           spatialArea=admin2Areas)

# Calculate general population totals at the subarea (Admin-2) x urban/rural
# level and using 1km resolution population grid. Assign urbanicity by
# thresholding population density based on estimated proportion population
# urban/rural, making sure total area (Admin-1) urban/rural populations in
# each area matches poppaKenya.
```

```
data(kenyaPopulationData)
pop.matKenya <- makePopIntegrationTab(
  km.res=5, pop=pop, domain.map.dat=adm0,
  east.lim=east.lim, north.lim=north.lim, map.projection=projKenya,
  poppa = poppaKenya, poppsub=poppsubKenya,
  area.map.dat = adm1, subarea.map.dat = adm2,
  areaNameVar = "NAME_1", subareaNameVar="NAME_2")

##### Now we make a model for the risk. We will use an SPDE model with these
##### parameters for the linear predictor on the logist scale, which are chosen
##### to be of practical interest:
beta0=-2.9 # intercept
gamma=-1 # urban effect
rho=(1/3)^2 # spatial variance
eff.range = 400 # effective spatial range in km
sigma.epsilon=sqrt(1/2.5) # cluster (nugget) effect standard deviation

# simulate the population! Note that this produces multiple dense
# nEA x nsim and nIntegrationPoint x nsim matrices. In the future
# sparse matrices will and chunk by chunk computations may be incorporated.
simPop = simPopSPDE(nsim=1, easpa=easpaKenyaNeonatal,
                    pop.mat=pop.matKenya, target.pop.mat=pop.matKenya,
                    poppsub=poppsubKenya, spde.mesh=kenyaMesh,
                    marg.var=rho, sigma.epsilon=sigma.epsilon,
                    gamma=gamma, eff.range=eff.range, beta0=beta0,
                    seed=123, inla.seed=12, n.HH.sampled=25,
                    stratify.by.urban=TRUE, subarea.level=TRUE,
                    do.fine.scale.risk=TRUE,
                    min1.per.subarea=TRUE)

pixelPop = simPop$pixelPop
subareaPop = pixelPopToArea(pixel.level.pop=pixelPop, ea.samples=pixelPop$ea.samples,
  areas=pop.matKenya$subarea, stratify.by.urban=TRUE,
  target.pop.mat=pop.matKenya, do.fine.scale.risk=TRUE)

# get areas associated with each subarea for aggregation
tempAreasFrom = pop.matKenya$subarea
tempAreasTo = pop.matKenya$area
areas.from = sort(unique(tempAreasFrom))
areas.toI = match(areas.from, tempAreasFrom)
areas.to = tempAreasTo[areas.toI]

# do the aggregation from subareas to areas
outAreaLevel = areaPopToArea(area.level.pop=subareaPop,
  areas.from=areas.from, areas.to=areas.to,
  stratify.by.urban=TRUE, do.fine.scale.risk=TRUE)

## End(Not run)
```

---

aggregateSurvey *Aggregate estimators from different surveys.*

Description

Aggregate estimators from different surveys.

Usage

```
aggregateSurvey(data)
```

Arguments

data	Output from getDirectList

Value

Estimators aggregated across surveys.

Author(s)

Zehang Richard Li

Examples

```
## Not run:
data(DemoData)
data(DemoMap)
years <- levels(DemoData[[1]]$time)

# obtain direct estimates
data <- getDirectList(births = DemoData,
years = years,
regionVar = "region", timeVar = "time",
clusterVar = "~clustid+id",
ageVar = "age", weightsVar = "weights",
geo.recode = NULL)

# obtain maps
geo <- DemoMap$geo
mat <- DemoMap$Amat

# Simulate hyper priors
priors <- simhyper(R = 2, nsamp = 1e+05, nsamp.check = 5000, Amat = mat, only.iid = TRUE)

# combine data from multiple surveys
data <- aggregateSurvey(data)
utils::head(data)


## End(Not run)
```

Benchmark *Benchmark posterior draws to national estimates*

Description

[Experimental]

Usage

```
Benchmark(
  fitted,
  national,
  estVar,
  sdVar,
  timeVar = NULL,
  weight.region = NULL,
  method = c("MH", "Rejection")[2]
)
```

Arguments

fitted	output from getSmoothed to be benchmarked.
national	a data frame of national level estimates that is benchmarked against, with at least two columns indicating national estimates (probability scale) and the associated standard error. If benchmarking over multiple time period, a third column indicating time period is needed.
estVar	column name in national that indicates national estimates.
sdVar	column name in national that indicates standard errors of national estimates.
timeVar	column name in national that indicates time periods.
weight.region	a data frame with a column region specifying subnational regions, a column proportion that specifies the proportion of population in each region. When multiple time periods exist, a third column years is required and the population proportions are the population proportions of each region in the corresponding time period.
method	a string denoting the algorithm to use for benchmarking. Options include MH for Metropolis-Hastings, and Rejection for rejection sampler. Defaults to Rejection.

Value

Benchmarked object in S3 class SUMMERproj or SUMMERprojlist in the same format as the input object fitted.

Author(s)

Taylor Okonek, Zehang Richard Li

References

Okonek, Taylor, and Jon Wakefield. "A computationally efficient approach to fully Bayesian benchmarking." Journal of Official Statistics 40, no. 2 (2024): 283-316.

Examples

```
## Not run:
##  ---------------------------------------- ##
##     Benchmarking with smoothCluster output
##  ---------------------------------------- ##

data(DemoData)
# fit unstratified cluster-level model
counts.all <- NULL
for(i in 1:length(DemoData)){
vars <- c("clustid", "region", "time", "age")
counts <- getCounts(DemoData[[i]][, c(vars, "died")],
variables = 'died',
      by = vars, drop=TRUE)
counts$cluster <- counts$clustid
counts$years <- counts$time
counts$Y <- counts$died
counts$survey <- names(DemoData)[i]
counts.all <- rbind(counts.all, counts)
}
periods <- c("85-89", "90-94", "95-99", "00-04", "05-09", "10-14", "15-19")
fit.bb  <- smoothCluster(data = counts.all, Amat = DemoMap$Amat,
family = "betabinomial",
year.label = periods,
survey.effect = TRUE)
est.bb <- getSmoothed(fit.bb, nsim = 1e4, CI = 0.95, save.draws=TRUE)

# construct a simple population weight data frame with equal weights
weight.region <- expand.grid(region = unique(counts.all$region),
 years = periods)
weight.region$proportion <- 0.25

# construct a simple national estimates
national <- data.frame(years = periods,
   est = seq(0.27, 0.1, length = 7),
   sd = runif(7, 0.01, 0.03))

 # benchmarking
est.bb.bench <- Benchmark(est.bb, national, weight.region = weight.region,
estVar = "est", sdVar = "sd", timeVar = "years")

# Sanity check: Benchmarking comparison
compare <- national
compare$before <- NA
compare$after <- NA
for(i in 1:dim(compare)[1]){
weights <- subset(weight.region, years == national$years[i])
```

```
sub <- subset(est.bb$overall, years == national$years[i])
sub <- merge(sub, weights)
sub.bench <- subset(est.bb.bench$overall, years == national$years[i])
sub.bench <- merge(sub.bench, weights)
compare$before[i] <- sum(sub$proportion * sub$median)
compare$after[i] <- sum(sub.bench$proportion * sub.bench$median)
}
plot(compare$est, compare$after, col = 2, pch = 10,
 xlim = range(c(compare$est, compare$before, compare$after)),
 ylim = range(c(compare$est, compare$before, compare$after)),
 xlab = "External national estimates",
 ylab = "Weighted posterior median after benchmarking",
    main = "Sanity check: weighted average of area medians")
points(compare$est, compare$before)
abline(c(0, 1))
legend("topleft", c("Before benchmarking", "After benchmarking"), pch = c(1, 10), col = c(1, 2))

#  construct a simple national estimates
national <- data.frame(years = periods,
   est = seq(0.22, 0.1, length = 7),
   sd = runif(7, 0.01, 0.03))
# national does not need to have all years
national_sub <- national[1:3,]

# benchmarking
est.bb.bench <- Benchmark(est.bb, national_sub,
weight.region = weight.region,
estVar = "est", sdVar = "sd", timeVar = "years")

# Sanity check: only benchmarked for three periods
compare <- national
compare$before <- NA
compare$after <- NA
for(i in 1:dim(compare)[1]){
weights <- subset(weight.region, years == national$years[i])
sub <- subset(est.bb$overall, years == national$years[i])
sub <- merge(sub, weights)
sub.bench <- subset(est.bb.bench$overall, years == national$years[i])
sub.bench <- merge(sub.bench, weights)
compare$before[i] <- sum(sub$proportion * sub$median)
compare$after[i] <- sum(sub.bench$proportion * sub.bench$median)
}
plot(compare$est, compare$after, col = 2, pch = 10,
 xlim = range(c(compare$est, compare$before, compare$after)),
 ylim = range(c(compare$est, compare$before, compare$after)),
 xlab = "External national estimates",
 ylab = "Weighted posterior median after benchmarking",
    main = "Sanity check: weighted average of area medians")
points(compare$est, compare$before)
abline(c(0, 1))
legend("topleft", c("Before benchmarking", "After benchmarking"), pch = c(1, 10), col = c(1, 2))

#  Another extreme benchmarking example, where almost all weights in central region
```

```
weight.region$proportion <- 0.01
weight.region$proportion[weight.region$region == "central"] <- 0.97
# benchmarking
est.bb.bench <- Benchmark(est.bb, national, weight.region = weight.region,
estVar = "est", sdVar = "sd", timeVar = "years")
# It can be seen the central region are pulled to the national benchmark
plot(national$est,
 subset(est.bb.bench$overall, region == "central")$mean,
 col = 2, pch = 10, xlab = "External national estimates",
 ylab = "Central region estimates")
points(national$est,
 subset(est.bb$overall, region == "central")$mean)
legend("topleft", c("Before benchmarking", "After benchmarking"), pch = c(1, 10),  col = c(1, 2))
abline(c(0, 1))

# Example with the MH method
# Benchmarking with MH should be applied when customized priors are
#  specified for fixed effects when fitting the model
fit.bb.new  <- smoothCluster(data = counts.all, Amat = DemoMap$Amat,
family = "betabinomial",
year.label = periods,
survey.effect = TRUE,
control.fixed = list(
mean=list(`age.intercept0:1`=-4,
       `age.intercept1-11:1`=-5,
       `age.intercept12-23:1`=-8,
       `age.intercept24-35:1`=-9,
       `age.intercept36-47:1`=-10,
       `age.intercept48-59:1`=-11),
prec=list(`age.intercept0:1`=10,
       `age.intercept1-11:1`=10,
       `age.intercept12-23:1`=10,
       `age.intercept24-35:1`=10,
       `age.intercept36-47:1`=10,
       `age.intercept48-59:1`=10)))
est.bb.new <- getSmoothed(fit.bb.new, nsim = 10000, CI = 0.95, save.draws=TRUE)

#  construct a simple national estimates
national <- data.frame(years = periods,
   est = seq(0.22, 0.1, length = 7),
   sd = runif(7, 0.01, 0.03))
weight.region <- expand.grid(region = unique(counts.all$region),
 years = periods)
weight.region$proportion <- 0.25
est.bb.bench.MH <- Benchmark(est.bb.new, national,
weight.region = weight.region,
estVar = "est", sdVar = "sd", timeVar = "years",
method = "MH")

compare <- national
compare$before <- NA
compare$after <- NA
for(i in 1:dim(compare)[1]){
```

```
weights <- subset(weight.region, years == national$years[i])
sub <- subset(est.bb.new$overall, years == national$years[i])
sub <- merge(sub, weights)
sub.bench <- subset(est.bb.bench.MH$overall, years == national$years[i])
sub.bench <- merge(sub.bench, weights)
compare$before[i] <- sum(sub$proportion * sub$median)
compare$after[i] <- sum(sub.bench$proportion * sub.bench$median)
}
plot(compare$est, compare$after, col = 2, pch = 10,
 xlim = range(c(compare$est, compare$before, compare$after)),
 ylim = range(c(compare$est, compare$before, compare$after)),
 xlab = "External national estimates",
 ylab = "Weighted posterior median after benchmarking",
    main = "Sanity check: weighted average of area medians")
points(compare$est, compare$before)
abline(c(0, 1))
legend("topleft", c("Before benchmarking", "After benchmarking"), pch = c(1, 10), col = c(1, 2))

##  ---------------------------------------- ##
##     Benchmarking with smoothDirect output
##  ---------------------------------------- ##
years <- levels(DemoData[[1]]$time)
# obtain direct estimates
data_multi <- getDirectList(births = DemoData, years = years,
                     regionVar = "region",  timeVar = "time", clusterVar = "~clustid+id",
                        ageVar = "age", weightsVar = "weights", geo.recode = NULL)
data <- aggregateSurvey(data_multi)
#  subnational model
years.all <- c(years, "15-19")
fit2 <- smoothDirect(data = data, Amat = DemoMap$Amat,
                 year.label = years.all, year.range = c(1985, 2019),
                 time.model = "rw2", m = 5, type.st = 4)
out2a <- getSmoothed(fit2, joint = TRUE, nsim = 1e5, save.draws = TRUE)

##
## Benchmarking for yearly estimates
##
weight.region <- expand.grid(region = unique(data$region[data$region != "All"]),
                             years = 1985:2019)
weight.region$proportion <- 0.25
# construct a simple national estimates
national <- data.frame(years = 1985:2019,
                       est = seq(0.25, 0.15, length = 35),
                       sd = runif(35, 0.03, 0.05))
# Benchmarking to national estimates on the yearly scale
out2b <- Benchmark(out2a, national, weight.region = weight.region,
                          estVar = "est", sdVar = "sd", timeVar = "years")
plot(out2a$overall)
plot(out2b$overall)

# combine the point estimate and compare with the benchmark values
national.est <- aggregate(mean ~ years,
   data = out2a$overall[out2a$overall$is.yearly, ], FUN = mean)
```

```
national.est.bench <- aggregate(mean ~ years,
   data = out2b$overall[out2b$overall$is.yearly, ], FUN = mean)

plot(national$est, national.est$mean,
 xlim = range(c(national$est, national.est$mean, national.est.bench$mean)),
 ylim = range(c(national$est, national.est$mean, national.est.bench$mean)),
 xlab = "External national estimates",
 ylab = "Weighted posterior median after benchmarking",
    main = "Sanity check: weighted average of area means")
points(national$est, national.est.bench$mean, col = 2, pch = 10)
abline(c(0, 1))
legend("topleft", c("Before benchmarking", "After benchmarking"), pch = c(1, 10), col = c(1, 2))



##
## Benchmarking for period estimates
##
weight.region <- expand.grid(region = unique(data$region[data$region != "All"]),
                             years = years.all)
weight.region$proportion <- 0.25
# construct a simple national estimates
national <- data.frame(years = years.all,
                       est = seq(0.25, 0.15, len = 7),
                       sd = runif(7, 0.01, 0.03))
# Benchmarking to national estimates on the period scale
out2c <- Benchmark(out2a, national, weight.region = weight.region,
                           estVar = "est", sdVar = "sd", timeVar = "years")
plot(out2a$overall)
plot(out2c$overall)

# combine the point estimate and compare with the benchmark values
national.est <- aggregate(mean ~ years,
data = out2a$overall[!out2a$overall$is.yearly, ], FUN = mean)
national.est.bench <- aggregate(mean ~ years,
data = out2c$overall[!out2b$overall$is.yearly, ], FUN = mean)

plot(national$est, national.est$mean,
 xlim = range(c(national$est, national.est$mean, national.est.bench$mean)),
 ylim = range(c(national$est, national.est$mean, national.est.bench$mean)),
 xlab = "External national estimates",
 ylab = "Weighted posterior median after benchmarking",
    main = "Sanity check: weighted average of area means")
points(national$est, national.est.bench$mean, col = 2, pch = 10)
abline(c(0, 1))
legend("topleft", c("Before benchmarking", "After benchmarking"), pch = c(1, 10), col = c(1, 2))



## End(Not run)
```

BRFSS *The BRFSS dataset*

Description

The Behavioral Risk Factor Surveillance System (BRFSS) is an annual telephone health survey conducted by the Centers for Disease Control and Prevention (CDC) that tracks health conditions and risk behaviors in the United States and its territories since 1984. This BRFSS dataset contains 16124 observations. The `diab2` variable is the binary indicator of Type II diabetes, `strata` is the strata indicator and `rwt_llcp` is the final design weight. Records with missing HRA code or diabetes status are removed from this dataset. See https://www.cdc.gov/brfss/annual_data/2013/pdf/Weighting_Data.pdf for more details of the weighting procedure.

Usage

```
data(BRFSS)
```

Format

A data.frame of 26 variables.

References

Washington State Department of Health, Center for Health Statistics. Behavioral Risk Factor Surveillance System, supported in part by the Centers for Disease Control and Prevention. Corporative Agreement U58/DP006066-01 (2015).

calibrateByRegion *Calibrate the point level totals so their sum matches the regional totals*

Description

Calibrate/normalize the point level totals so their sum matches the regional totals. Technically, the totals can be at any level smaller than the region level specified.

Usage

```
calibrateByRegion(point.totals, point.regions, regions, region.totals)
```

Arguments

`point.totals`	Vector of point level totals that will be calibrated/normalized
`point.regions`	Vector of regions associated with each point
`regions`	Vector of region names
`region.totals`	Vector of desired region level totals associated with `regions`

Details

[Experimental]

Value

A vector of same length as point.totals and point.regions containing the calibrated/normalized point totals that sum to the correct regional totals

Vector of updated point level totals, calibrated to match region totals

Author(s)

John Paige

Examples

```
point.totals = c(1, 1, 1, 2)
point.regions = c("a", "a", "b", "b")
region.totals = c(10, 20)
regions = c("a", "b")
calibrateByRegion(point.totals, point.regions, regions, region.totals)
```

---

changeRegion *Map region names to a common set.*

Description

Map region names to a common set.

Usage

```
changeRegion(data, Bmat, regionVar = "region")
```

Arguments

data	Preprocessed data
Bmat	Matrix of changes. Each row corresponds to a region name possibly in the data files, and each column corresponds to a region after mapping. The values in the matrix are binary. The row names and column names need to be specified to the region names.
regionVar	String indicating the region variable. Defaults to 'region'.

Value

Data after changing region names

Author(s)

Zehang Richard Li

Examples

```
# Construct a small test data
testdata <- data.frame(region = c("north", "south", "east",
 "south", "east"), index = c(1:5))

# Construct a changing rule: combining south and east
Bmat <- matrix(c(1, 0, 0, 0, 1, 1), 3, 2)
colnames(Bmat) <- c("north", "south and east")
rownames(Bmat) <- c("north", "south", "east")
print(Bmat)

# New data after transformation
test <- changeRegion(testdata, Bmat, "region")
print(test)
```

---

compareEstimates *Plot heatmap comparing pairwise posterior exceedence probabilities for svysae object*

Description

Plot heatmap comparing pairwise posterior exceedence probabilities for svysae object

Usage

```
compareEstimates(x, posterior.sample = NULL, title = NULL, return.plot = FALSE)
```

Arguments

x	an object in the S3 class of svysae, fhModel, or clusterModel. Plots are created for all models in this object.
posterior.sample	Matrix of posteriors samples of area level quantities with one row for each area and one column for each sample. This argument may be specified to only provide a heatmap for the desired samples.
title	Optional parameter changing the title of the plot
return.plot	Logical indicator for whether the ggplot object is returned

Value

ggplot containing heat map of pairwise comparisons

Examples

```
## Not run:
data(DemoData2)
data(DemoMap2)
library(survey)
des0 <- svydesign(ids = ~clustid+id, strata = ~strata,
                  weights = ~weights, data = DemoData2, nest = TRUE)
Xmat <- aggregate(age~region, data = DemoData2, FUN = mean)

cts.res <- smoothArea(tobacco.use ~ 1,
                      domain = ~region,
                      design = des0,
                      adj.mat = DemoMap2$Amat,
                      pc.u = 1,
                      pc.alpha = 0.01,
                      pc.u.phi = 0.5,
                      pc.alpha.phi = 2/3,
                      return.samples = TRUE)
compareEstimates(cts.res)

## End(Not run)
```

DemoData *Simulated child mortality person-month dataset.*

Description

A small simulated dataset with 4 regions and 5 survey years. This does not represent any real country's data and are based on a subset of the model dataset provided by DHS.

Usage

```
data(DemoData)
```

Format

A list of with five components, named by survey year.

Source

https://dhsprogram.com/data/model-datasets.cfm

DemoData2 *Simulated dataset for prevalence mapping.*

Description

A small fake dataset with 8 regions and two response variables: age and tobacco.use. This does not represent any real country's data and are based on a subset of the model dataset provided by DHS.

Usage

```
data(DemoData2)
```

Format

A data.frame of 7 variables.

Source

https://dhsprogram.com/data/model-datasets.cfm

DemoMap *Uganda Admin-1 region map for illustration purpose*

Description

Shapefiles are from 1995 Uganda Admin 1 regions provided by DHS, but the data do not represent real information about any country.

Usage

```
data(DemoMap)
```

Format

An object of class `list` of length 2.

Details

- geo. Geographic map files
- Amat. Adjacency matrix for regions

Source

https://spatialdata.dhsprogram.com/boundaries/#view=table&countryId=UG

DemoMap2 *Kenya Admin-1 region map for illustration purpose*

Description

Shapefiles are from 2014 Kenya Admin 1 regions provided by DHS.

Usage

```
data(DemoMap2)
```

Format

An object of class `list` of length 2.

Details

- geo Geographic map files
- Amat Adjacency matrix for regions

Source

https://spatialdata.dhsprogram.com/boundaries/#view=table&countryId=KE

expit *Expit transformation*

Description

Expit transformation

Usage

```
expit(x)
```

Arguments

x	data

Value

expit of x

Examples

```
x <- .5
expit(x)
```

getAdjusted *Adjust direct estimates and their associated variances*

Description

Adjust direct estimates and their associated variances

Usage

```
getAdjusted(
  data,
  ratio,
  time = "years",
  region = "region",
  est = "mean",
  logit = "logit.est",
  logit.var = "var.est",
  logit.prec = "logit.prec",
  logit.lower = "lower",
  logit.upper = "upper",
  prob.lower = NULL,
  prob.upper = NULL,
  adj = "ratio",
  verbose = FALSE,
  lower = NULL,
  upper = NULL
)
```

Arguments

data	data frame of the adjusted estimates and the associated uncertainties, see the arguments below for specific columns.
ratio	the ratio of unadjusted mortality rates to the true mortality rates. It can be either a data frame with the following three columns (region, time, and adj) if adjustment factor differ by region; or a data frame with the following two columns (time and adj) if adjustment factor only varies over time. The column names specifying region, time, and adjustment are specified by the arguments in the function call.
time	the column name for time in the data and adjustment ratio.
region	the column name for region in the data and adjustment ratio.
est	the column name for unadjusted mortality rates in the data
logit	the column name for the logit of the unadjusted mortality rates in the data
logit.var	the column name for the variance of the logit of the unadjusted mortality rates in the data
logit.prec	the column name for the precision of the logit of the unadjusted mortality rates in the data

logit.lower	the column name for the 95% lower bound of the logit of the unadjusted mortality rates in the data
logit.upper	the column name for the 95% lower bound of the logit of the unadjusted mortality rates in the data
prob.lower	the column name for the 95% lower bound of the unadjusted mortality rates in the data. If this is provided instead of logit.lower, the logit scale lower bound will be created.
prob.upper	the column name for the 95% lower bound of the unadjusted mortality rates in the data. if this is provided instead of logit.upper, the logit scale upper bound will be created.
adj	the column name for the adjustment ratio
verbose	logical indicator for whether to print out unadjusted row index
lower	previous argument name for prob.lower. Will be removed in the next update
upper	previous argument name for prob.upper. Will be removed in the next update

Value

adjusted dataset of the same columns.

Author(s)

Zehang Richard Li

Examples

```
## Not run:
years <- levels(DemoData[[1]]$time)

# obtain direct estimates
data <- getDirectList(births = DemoData,
years = years,
regionVar = "region", timeVar = "time",
clusterVar = "~clustid+id",
ageVar = "age", weightsVar = "weights",
geo.recode = NULL)
# obtain direct estimates
data_multi <- getDirectList(births = DemoData, years = years,
  regionVar = "region",  timeVar = "time", clusterVar = "~clustid+id",
  ageVar = "age", weightsVar = "weights", geo.recode = NULL)
data <- aggregateSurvey(data_multi)

# randomly simulate adjustment factor
adj <- expand.grid(region = unique(data$region), years = years)
adj$ratio <- runif(dim(adj)[1], min = 0.5, max = 0.8)
data.adj <- getAdjusted(data = data, ratio = adj)

## End(Not run)
```

getAmat *Extract adjacency matrix from the map*

Description

Extract adjacency matrix from the map

Usage

```
getAmat(geo, names)
```

Arguments

geo	SpatialPolygonsDataFrame of the map
names	character vector of region ids to be added to the neighbours list

Value

Spatial djacency matrix.

Author(s)

Zehang Richard Li

Examples

```
## Not run:
data(DemoMap)
mat <- getAmat(geo = DemoMap$geo, names = DemoMap$geo$REGNAME)
mat
DemoMap$Amat

## End(Not run)
```

getAreaName *Determines which administrative areas contain the given points*

Description

[Experimental]

Usage

```
getAreaName(
  pts,
  shapefile,
  areaNameVar = "NAME_1",
  delta = 0.05,
  mean.neighbor = 50,
  max.bytes = 3 * 2^30
)
```

Arguments

pts	2 column matrix of lon/lat coordinates
shapefile	A SpatialPolygonsDataFrame object
areaNameVar	The column name in `slot(shapefile, "data")` corresponding to the area level of interest
delta	Argument passed to fields::fields.rdist.near in fields package
mean.neighbor	Argument passed to fields::fields.rdist.near in fields package
max.bytes	Maximum allowed memory in bytes (default is 3Gb). Determines whether to call fields::fields.rdist.near which saves memory but requires delta and mean.neighbor inputs to be specified for fields::fields.rdist.near

Details

For any points not in an area, they are assigned the nearest area using fields::fields.rdist.near or fields::rdist depending on the number of points and the maximum memory in bytes with a warning.

delta and mean.neighbor arguments only used when some points are not in areas, perhaps due to inconsistencies in shapefiles.

Value

A list of area IDs, area names, whether or not points are in multiple areas, and whether or not points are in no areas and assigned to the nearest one.

Author(s)

John Paige

See Also

projKenya, fields.rdist.near

Examples

```
## Not run:
# download Kenya GADM shapefiles from SUMMERdata github repository
githubURL <- "https://github.com/paigejo/SUMMERdata/blob/main/data/kenyaMaps.rda?raw=true"
download.file(githubURL,"kenyaMaps.rda")
```

```
# load it in
load("kenyaMaps.rda")

# use the shapefile data to see what Admin1 and 2 areas the
# points (0, 37) and (0.5, 38) are in
# (these are longitude/latitude coordinates)
pts = cbind(c(37, 38), c(0, .5))
head(slot(adm1, "data"))
admin1Areas = getAreaName(pts, adm1, "NAME_1")
admin2Areas = getAreaName(pts, adm2, "NAME_2")

## End(Not run)
```

---

getBirths *Reformat full birth records into person-month format*

Description

Reformat full birth records into person-month format

Usage

```
getBirths(
  filepath = NULL,
  data = NULL,
  surveyyear = NA,
 variables = c("caseid", "v001", "v002", "v004", "v005", "v021", "v022", "v023", "v024",
    "v025", "v139", "bidx"),
  strata = c("v024", "v025"),
  dob = "b3",
  alive = "b5",
  age = "b7",
  age.truncate = 24,
  date.interview = "v008",
  month.cut = c(1, 12, 24, 36, 48, 60),
  year.cut = seq(1980, 2020, by = 5),
  min.last.period = 0,
  cmc.adjust = 0,
  compact = FALSE,
  compact.by = c("v001", "v024", "v025", "v005")
)
```

Arguments

filepath	file path of raw .dta file from DHS. Only used when data frame is not provided in the function call.

data	data frame of a DHS survey
surveyyear	year of survey. Observations after this year will be excluded from the analysis.
variables	vector of variables to be used in obtaining the person-month files. The variables correspond the the DHS recode manual VI. For early DHS data, the variable names may need to be changed.
strata	vector of variable names used for strata. If a single variable is specified, then that variable will be used as strata indicator If multiple variables are specified, the interaction of these variables will be used as strata indicator.
dob	variable name for the date of birth.
alive	variable name for the indicator of whether child was alive or dead at the time of interview. It should be factor or character variable with levels "no" or "yes". Other coding scheme will not be recognized and can lead to errors.
age	variable name for the age at death of the child in completed months.
age.truncate	the smallest age in months where only full years are reported. The default value is 24, which corresponds to the DHS practice of recording only age in full years for children over 2 years old. That is, for children with age starting from 24 months old, we assume the age variable reported in multiples of 12 are truncated from its true value. For example, children between age 24 to 35 months are all recorded as 24. To account for the truncation of age, 5 months are added to all ages recorded in multiples of 12 starting from 24. To avoid this adjustment, set this argument to NA.
date.interview	variable name for the date of interview.
month.cut	the cutoff of each bins of age group in the unit of months. Default values are 1, 12, 24, 36, 48, and 60, representing the age groups (0, 1), [1, 12), [12, 24), ..., [48, 60).
year.cut	The cutoff of each bins of time periods, including both boundaries. Default values are 1980, 1985, ..., 2020, representing the time periods 80-84, 85-89, ..., 15-19. Notice that if each bin contains one year, the last year in the output is max(year.cut)-1. For example, if year.cut = 1980:2020, the last year in the output is 2019.
min.last.period	The cutoff for how many years the last period must contain in order to be counted in the output. For example, if the last period is 2015-2019 and min.last.period = 3, person-months for the last period will only be returned if survey contains observations at least in 2017. This argument avoids the situation that estimates for the last period being based on only a small number of initial years, if applicable. Default to be 0.
cmc.adjust	number of months to add to the recorded month in the dataset. Some DHS surveys does not use Gregorian calendar (the calendar used in most of the world). For example, the Ethiopian calendar is 92 months behind the Gregorian calendar in general. Then we can set cmc.adjust to 92, which adds 92 months to all dates in the dataset, effectively transforming the Ethiopian calendar to the Gregorian calendar.
compact	logical indicator of whether the compact format is returned. In the compact output, person months are aggregated by cluster, age, and time. Total number of

person months and deaths in each group are returned instead of the raw person-months.

`compact.by` vector of variables to summarize the compact form by.

Value

This function returns a new data frame where each row indicate a person-month, with the additional variables specified in the function argument.

Author(s)

Zehang Richard Li, Bryan Martin, Laina Mercer

References

Li, Z., Hsiao, Y., Godwin, J., Martin, B. D., Wakefield, J., Clark, S. J., & with support from the United Nations Inter-agency Group for Child Mortality Estimation and its technical advisory group. (2019). *Changes in the spatial distribution of the under-five mortality rate: Small-area analysis of 122 DHS surveys in 262 subregions of 35 countries in Africa.* PloS one, 14(1), e0210645.

Mercer, L. D., Wakefield, J., Pantazis, A., Lutambi, A. M., Masanja, H., & Clark, S. (2015). *Space-time smoothing of complex survey data: small area estimation for child mortality.* The annals of applied statistics, 9(4), 1889.

Examples

```
## Not run:
# my_fp <- "/myExampleFilepath/surveyData.DTA"
# DemoData <- getBirths(filepath = my_fp, surveyyear = 2015)

## End(Not run)
```

`getCounts` *Aggregate person-month data into counts and totals by groups.*

Description

Aggregate person-month data into counts and totals by groups.

Usage

```
getCounts(data, variables, by, ignore = NULL, addtotal = TRUE, drop = TRUE)
```

Arguments

data	dataset in person-month format
variables	a character vector of the variables to aggregate
by	a character vector of columns that specifies which groups to aggregate by.
ignore	list of conditions not to impute 0. If left unspecified, any group levels not in the data will be imputed to have 0 counts.
addtotal	logical indicator of whether to add a column of group total counts.
drop	logical indicator of whether to drop all rows with total = 0.

Value

data.frame of the ggregated counts.

Author(s)

Zehang Richard Li

Examples

```
# a toy dataset with 4 time periods but one missing in data
timelist <- factor(1:4)
data = data.frame(died = c(0,0,0,1,1,0,0),
area = c(rep(c("A", "B"), 3), "A"),
time = timelist[c(1,1,2,3,3,3,3)])
data
# without ignore argument, all levels will be imputed
getCounts(data, variables = "died", by = c("area", "time"))

# ignoring time = 4, the ignored level will not be imputed (but still in the output)
getCounts(data, variables = "died", by = c("area", "time"),
ignore = list("time"=c(4)) )
```

---

getDiag *Extract posterior summaries of random effects*

Description

Extract posterior summaries of random effects

Usage

```
getDiag(
  fitted,
  inla_mod = deprecated(),
  field = c("space", "time", "spacetime")[1],
  CI = 0.95,
  draws = NULL,
  nsim = 1000,
  ...
)
```

Arguments

fitted	output from smoothDirect or smoothCluster
inla_mod	**[Deprecated]** replaced by fitted.
field	which random effects to plot. It can be one of the following: space, time, and spacetime.
CI	Desired level of credible intervals
draws	Posterior samples drawn from the fitted model. This argument allows the previously sampled draws (by setting save.draws to be TRUE) be used in new aggregation tasks.
nsim	number of simulations, only applicable for the cluster-level model space-time interaction terms when random slopes are included.
...	Unused arguments, for users with fitted object from the package before v1.0.0, arguments including Amat, year.label, and year.range can still be specified manually.

Value

List of diagnostic plots

Author(s)

Zehang Richard Li

Examples

```
## Not run:
  data(DemoMap)
  years <- levels(DemoData[[1]]$time)

  # obtain direct estimates
  data <- getDirectList(births = DemoData,
  years = years,
  regionVar = "region", timeVar = "time",
  clusterVar = "~clustid+id",
  ageVar = "age", weightsVar = "weights",
  geo.recode = NULL)
```

```
  # obtain direct estimates
  data_multi <- getDirectList(births = DemoData, years = years,
    regionVar = "region",  timeVar = "time", clusterVar = "~clustid+id",
    ageVar = "age", weightsVar = "weights", geo.recode = NULL)
  data <- aggregateSurvey(data_multi)

  #  national model
  years.all <- c(years, "15-19")
  fit1 <- smoothDirect(data = data, geo = DemoMap$geo, Amat = DemoMap$Amat,
    year.label = years.all, year.range = c(1985, 2019),
    rw = 2, is.yearly=FALSE, m = 5)
random.time <- getDiag(fit1, field = "time")
  random.space <- getDiag(fit1, field = "space")
  random.spacetime <- getDiag(fit1, field = "spacetime")

## End(Not run)
```

`getDirect` *Obtain the Horvitz-Thompson direct estimates and standard errors using delta method for a single survey.*

Description

Obtain the Horvitz-Thompson direct estimates and standard errors using delta method for a single survey.

Usage

```
getDirect(
  births,
  years,
  regionVar = "region",
  timeVar = "time",
  clusterVar = "~v001+v002",
  ageVar = "age",
  weightsVar = "v005",
  Ntrials = NULL,
  geo.recode = NULL,
  national.only = FALSE,
  CI = 0.95
)
```

Arguments

`births`	A matrix child-month data from `getBirths`
`years`	String vector of the year intervals used
`regionVar`	Variable name for region in the input births data.

timeVar	Variable name for the time period indicator in the input births data.
clusterVar	Variable name for cluster, typically '~v001 + v002'
ageVar	Variable name for age group. This variable need to be in the form of "a-b" where a and b are both ages in months. For example, "1-11" means age between 1 and 11 months, including both end points. An exception is age less than one month can be represented by "0" or "0-0".
weightsVar	Variable name for sampling weights, typically 'v005'
Ntrials	Variable for the total number of person-months if the input data (births) is in the compact form.
geo.recode	The recode matrix to be used if region name is not consistent across different surveys. See changeRegion.
national.only	Logical indicator to obtain only the national estimates
CI	the desired confidence interval to calculate

Value

a matrix of period-region summary of the Horvitz-Thompson direct estimates by region and time period specified in the argument, the standard errors using delta method for a single survey, the 95% confidence interval, and the logit of the estimates.

Author(s)

Zehang Richard Li, Bryan Martin, Laina Mercer

References

Li, Z., Hsiao, Y., Godwin, J., Martin, B. D., Wakefield, J., Clark, S. J., & with support from the United Nations Inter-agency Group for Child Mortality Estimation and its technical advisory group. (2019). *Changes in the spatial distribution of the under-five mortality rate: Small-area analysis of 122 DHS surveys in 262 subregions of 35 countries in Africa.* PloS one, 14(1), e0210645.

Mercer, L. D., Wakefield, J., Pantazis, A., Lutambi, A. M., Masanja, H., & Clark, S. (2015). *Space-time smoothing of complex survey data: small area estimation for child mortality.* The annals of applied statistics, 9(4), 1889.

See Also

getDirectList

Examples

```
## Not run:
data(DemoData)
years <- c("85-89", "90-94", "95-99", "00-04", "05-09", "10-14")
mean <- getDirect(births = DemoData[[1]],  years = years,
regionVar = "region", timeVar = "time", clusterVar = "~clustid+id",
ageVar = "age", weightsVar = "weights", geo.recode = NULL)

## End(Not run)
```

getDirectList *Obtain the Horvitz-Thompson direct estimates and standard errors using delta method for multiple surveys.*

Description

Obtain the Horvitz-Thompson direct estimates and standard errors using delta method for multiple surveys.

Usage

```
getDirectList(
  births,
  years,
  regionVar = "region",
  timeVar = "time",
  clusterVar = "~v001+v002",
  ageVar = "age",
  weightsVar = "v005",
  Ntrials = NULL,
  geo.recode = NULL,
  national.only = FALSE
)
```

Arguments

births	A list of child-month data from multiple surveys from getBirths. The name of the list is used as the identifier in the output.
years	String vector of the year intervals used
regionVar	Variable name for region, typically 'v024', for older surveys might be 'v101'
timeVar	Variable name for the time period indicator in the input births data.
clusterVar	Variable name for the IDs in the second-stage cluster sampling, typically '~v001 + v002', i.e., the cluster number and household number. When no cluster sampling design exists, this variable usually is the household ID.
ageVar	Variable name for age group. This variable need to be in the form of "a-b" where a and b are both ages in months. For example, "1-11" means age between 1 and 11 months, including both end points. An exception is age less than one month can be represented by "0" or "0-0".
weightsVar	Variable name for sampling weights, typically 'v005'
Ntrials	Variable for the total number of person-months if the input data (births) is in the compact form.
geo.recode	The recode matrix to be used if region name is not consistent across different surveys. See changeRegion.
national.only	Logical indicator to obtain only the national estimates

Value

This is the extension to the getDirect function that returns estimates from multiple surveys. Additional columns in the output (survey and surveyYears) specify the estimates from different surveys.

Author(s)

Zehang Richard Li, Bryan Martin, Laina Mercer

References

Li, Z., Hsiao, Y., Godwin, J., Martin, B. D., Wakefield, J., Clark, S. J., & with support from the United Nations Inter-agency Group for Child Mortality Estimation and its technical advisory group. (2019). *Changes in the spatial distribution of the under-five mortality rate: Small-area analysis of 122 DHS surveys in 262 subregions of 35 countries in Africa.* PloS one, 14(1), e0210645.

Mercer, L. D., Wakefield, J., Pantazis, A., Lutambi, A. M., Masanja, H., & Clark, S. (2015). *Space-time smoothing of complex survey data: small area estimation for child mortality.* The annals of applied statistics, 9(4), 1889.

See Also

getDirect

Examples

```
## Not run:
data(DemoData)
years <- c("85-89", "90-94", "95-99", "00-04", "05-09", "10-14")
mean <- getDirectList(births = DemoData, years = years,
regionVar = "region", timeVar = "time", clusterVar = "~clustid+id",
ageVar = "age", weightsVar = "weights", geo.recode = NULL)

## End(Not run)
```

---

getSmoothed *Extract smoothed estimates.*

Description

Extract smoothed estimates.

Usage

```
getSmoothed(
  fitted,
  inla_mod = deprecated(),
  nsim = 1000,
  weight.strata = NULL,
  weight.frame = NULL,
```

```
  verbose = FALSE,
  mc = 0,
  include.time.unstruct = FALSE,
  include_time_unstruct = deprecated(),
  CI = 0.95,
  draws = NULL,
  save.draws = FALSE,
  include.subnational = TRUE,
  include_subnational = deprecated(),
  only.age = NULL,
  joint = FALSE,
  ...
)
```

Arguments

fitted	output from smoothDirect or smoothCluster
inla_mod	**[Deprecated]** replaced by fitted
nsim	number of simulations, only applicable for the cluster-level model or direct model when joint = TRUE is specified. The smooth direct model will draw 1e5 samples from the marginal distribution when joint = FALSE since the computation is faster.
weight.strata	a data frame with two columns specifying time and region, followed by columns specifying proportion of each strata for each region. This argument specifies the weights for strata-specific estimates on the probability scale.
weight.frame	a data frame with three columns, years, region, and the weight of each frame for the corresponding time period and region. This argument specifies the weights for frame-specific estimates on the logit scale. Notice this is different from weight.strata argument.
verbose	logical indicator whether to print progress messages from inla.posterior.sample.
mc	number of monte carlo draws to approximate the marginal prevalence/hazards for binomial model. If mc = 0, analytical approximation is used. The analytical approximation is invalid for hazard modeling with more than one age groups.
include.time.unstruct	Indicator whether to include the temporal unstructured effects (i.e., shocks) in the smoothed estimates from cluster-level model. The argument only applies to the cluster-level models (from smoothCluster). Default is FALSE which excludes all unstructured temporal components. If set to TRUE all the unstructured temporal random effects will be included. Alternatively, if this is specified as a vector of subset of year labels (as in the year.label argument), only the unstructured terms in the corresponding time periods will be added to the prediction.
include_time_unstruct	**[Deprecated]** replaced by include.time.unstruct
CI	Desired level of credible intervals
draws	Posterior samples drawn from the fitted model. This argument allows the previously sampled draws (by setting save.draws to be TRUE) be used in new aggregation tasks.

save.draws Logical indicator whether the raw posterior draws will be saved. Saved draws can be used to accelerate aggregations with different weights.

include.subnational
logical indicator whether to include the spatial and space-time interaction components in the smoothed estimates. If set to FALSE, only the main temporal trends are returned.

include_subnational
[Deprecated] replaced by include.subnational

only.age a vector of age groups used to compute the final estimates. Default to be NULL, which includes all age groups in the model. This argument can be used to extract mortality rates of finer age groups when fitting multiple age groups jointly.

joint Logical indicator whether the posterior draws should be drawn from the joint posterior or marginal distributions. Only releveant for the smooth direct model. Default from the marginal distribution (joint = FALSE).

... Unused arguments, for users with fitted object from the package before v1.0.0, arguments including Amat, year.label, and year.range can still be specified manually.

Value

A data frame or a list of data frames of S3 class SUMMERproj, which contains the smoothed estimates.

Author(s)

Zehang Richard Li

See Also

plot.SUMMERproj

Examples

```
## Not run:
years <- levels(DemoData[[1]]$time)

# obtain direct estimates
data <- getDirectList(births = DemoData,
years = years,
regionVar = "region", timeVar = "time",
clusterVar = "~clustid+id",
ageVar = "age", weightsVar = "weights",
geo.recode = NULL)
# obtain direct estimates
data_multi <- getDirectList(births = DemoData, years = years,
  regionVar = "region",  timeVar = "time", clusterVar = "~clustid+id",
  ageVar = "age", weightsVar = "weights", geo.recode = NULL)
data <- aggregateSurvey(data_multi)

##
```

```
## The following example shows extracting estimates from
##   fitted smoothDirect() model
##
#  national model
years.all <- c(years, "15-19")
fit1 <- smoothDirect(data = data, Amat = NULL,
  year.label = years.all, year.range = c(1985, 2019),
  rw = 2, is.yearly=FALSE, m = 5)
out1 <- getSmoothed(fit1)
plot(out1, is.subnational=FALSE)

#  subnational model
fit2 <- smoothDirect(data = data, Amat = mat,
  year.label = years.all, year.range = c(1985, 2019),
  rw = 2, is.yearly=TRUE, m = 5, type.st = 4)
out2 <- getSmoothed(fit2)
plot(out2, is.yearly=TRUE, is.subnational=TRUE)

## The following examples compare the marginal posterior draws
##    with joint posterior draws. The latter gives draws of
##    all estimates in addition to marginal summaries.
## The plots should fall closely along the 45 degree line

#  national period model
years.all <- c(years, "15-19")
fit0 <- smoothDirect(data = data, Amat = NULL,
year.label = years, year.range = c(1985, 2019),
time.model = 'rw2', m = 1, control.compute = list(config =TRUE))
out0 <- getSmoothed(fit0)
out0a <- getSmoothed(fit0, joint = TRUE, nsim = 1e5)
par(mfrow = c(1, 3))
plot(out0$median, out0a$overall$median)
abline(c(0, 1))
plot(out0$upper, out0a$overall$upper)
abline(c(0, 1))
plot(out0$lower, out0a$overall$lower)
abline(c(0, 1))

#  national yearly model
years.all <- c(years, "15-19")
fit1 <- smoothDirect(data = data, Amat = NULL,
year.label = years.all, year.range = c(1985, 2019),
time.model = 'rw2', m = 5, control.compute = list(config =TRUE))
out1 <- getSmoothed(fit1)
out1a <- getSmoothed(fit1, joint = TRUE, nsim = 1e5, save.draws = TRUE)
par(mfrow = c(1, 3))
plot(out1$median, out1a$overall$median)
abline(c(0, 1))
plot(out1$upper, out1a$overall$upper)
abline(c(0, 1))
plot(out1$lower, out1a$overall$lower)
abline(c(0, 1))
```

```
#  subnational model
fit2 <- smoothDirect(data = data, Amat = DemoMap$Amat,
year.label = years.all, year.range = c(1985, 2019),
time.model = 'rw2',is.yearly=TRUE, m = 5, type.st = 4,
control.compute = list(config =TRUE))
out2 <- getSmoothed(fit2)
out2a <- getSmoothed(fit2, joint = TRUE, nsim = 1e5, save.draws = TRUE)
par(mfrow = c(1, 3))
plot(out2$median, out2a$overall$median)
abline(c(0, 1))
plot(out2$upper, out2a$overall$upper)
abline(c(0, 1))
plot(out2$lower, out2a$overall$lower)
abline(c(0, 1))

#  subnational space-only model combining all periods
fit3 <- smoothDirect(data = data,
       time.model = NULL, Amat = DemoMap$Amat,
   control.compute = list(config =TRUE))
out3 <- getSmoothed(fit3)
out3a <- getSmoothed(fit3, joint = TRUE, nsim = 1e5, save.draws = TRUE)
par(mfrow = c(1, 3))
plot(out3$median, out3a$overall$median)
abline(c(0, 1))
plot(out3$upper, out3a$overall$upper)
abline(c(0, 1))
plot(out3$lower#' , out3a$overall$lower)
abline(c(0, 1))

## End(Not run)
```

`hatchPlot` *Plot maps with uncertainty hatching.*

Description

This function visualizes the map with different variables. The input data frame can be either the long or wide format.

Usage

```
hatchPlot(
  data,
  variables,
  values = NULL,
  labels = NULL,
  geo,
  by.data,
```

```
  by.geo,
  is.long = FALSE,
  lower,
  upper,
  lim = NULL,
  lim.CI = NULL,
  breaks.CI = NULL,
  ncol = 4,
  hatch = NULL,
  border = NULL,
  size = 1,
  legend.label = NULL,
  per1000 = FALSE,
  direction = 1,
  ...
)
```

Arguments

data	a data frame with variables to be plotted
variables	vector of variables to be plotted. If long format of data is used, only one variable can be selected
values	the column corresponding to the values to be plotted, only used when long format of data is used
labels	vector of labels to use for each variable, only used when wide format of data is used
geo	SpatialPolygonsDataFrame object for the map
by.data	column name specifying region names in the data
by.geo	variable name specifying region names in the data
is.long	logical indicator of whether the data is in the long format, default to FALSE
lower	column name of the lower bound of the CI
upper	column name of the upper bound of the CI
lim	fixed range of values for the variables to plot
lim.CI	fixed range of the CI widths to plot
breaks.CI	a vector of numerical values that decides the breaks in the CI widths to be shown
ncol	number of columns for the output tabs
hatch	color of the hatching lines.
border	color of the polygon borders.
size	line width of the polygon borders.
legend.label	Label for the color legend.
per1000	logical indicator to plot mortality rates as rates per 1,000 live births. Note that the added comparison data should always be in the probability scale.
direction	Direction of the color scheme. It can be either 1 (smaller values are darker) or -1 (higher values are darker). Default is set to 1.
...	unused.

Author(s)

Zehang Richard Li, Katie Wilson

Examples

```
## Not run:
years <- levels(DemoData[[1]]$time)

# obtain direct estimates
data <- getDirectList(births = DemoData,
years = years,
regionVar = "region", timeVar = "time",
clusterVar = "~clustid+id",
ageVar = "age", weightsVar = "weights",
geo.recode = NULL)
# obtain direct estimates
data_multi <- getDirectList(births = DemoData, years = years,
  regionVar = "region",  timeVar = "time", clusterVar = "~clustid+id",
  ageVar = "age", weightsVar = "weights", geo.recode = NULL)
data <- aggregateSurvey(data_multi)

fit2 <- smoothDirect(data = data, geo = geo, Amat = mat,
  year.label = years.all, year.range = c(1985, 2019),
  rw = 2, is.yearly=TRUE, m = 5, type.st = 4)
out2 <- getSmoothed(fit2)

plot(out2, is.yearly=TRUE, is.subnational=TRUE)

hatchPlot(data = subset(out2, is.yearly==FALSE), geo = geo,
variables=c("years"), values = c("median"),
by.data = "region", by.geo = "REGNAME",
lower = "lower", upper = "upper", is.long=TRUE)


## End(Not run)
```

`iid.new` *New random IID models for m-year to period random effects*

Description

New random IID models for m-year to period random effects

Usage

```
iid.new(
  cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
  theta = NULL
)
```

Arguments

cmd	list of model components
theta	log precision

---

iid.new.pc *New random IID models for m-year to period random effects*

Description

New random IID models for m-year to period random effects

Usage

```
iid.new.pc(
  cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
  theta = NULL
)
```

Arguments

cmd	list of model components
theta	log precision

KenData *Auxiliary data for Kenya 2014 DHS.*

Description

The list contains several data frames.

Usage

```
data(KenData)
```

Format

An object of class list of length 4.

Details

- HIV2014, a data frame with three columns: years (in five year periods), region (8 Admin-1 region groups), and the estimated bias of the reported U5MR due to HIV for each 5 year period from 1990-1994 to 2010-2014. The bias is represented as the ratio of the reported U5MR to the true U5MR.
- HIV2014.yearly, a data frame with three columns: years (in one year interval), region (8 Admin-1 region groups), and the estimated bias of the reported U5MR due to HIV for each year from 1980 to 2014. The bias is represented as the ratio of the reported U5MR to the true U5MR.
- IGME2019. Yearly Estimates of national under-5 child mortality in Kenya from the 2019 UN-IGME estimates.
- UrbanProp. Proportion of urban population by county and total population by county. Source: 2009 Kenya Population and Housing Census, and Table A2 of Kenya 2014 DHS report.

References

Neff Walker, Kenneth Hill, and Fengmin Zhao (2012) *Child mortality estimation: methods used to adjust for bias due to aids in estimating trends in under-five mortality.*,
PLoS Medicine, 9(8):e1001298.

`kenyaPopulationData` *Kenya 2009 Census Frame and Related Datasets*

Description

Datasets related to the 2009 census frame for Kenya based on the 2009 Kenya Population and Housing Census. General population totals are estimated for 2014. Based on 2014 population density estimates interpolated with exponential growth rate between 2010 and 2015 from WorldPop data.

Usage

```
data(kenyaPopulationData)

easpaKenyaNeonatal

poppaKenya

poppsubKenya
```

Format

A number of data.frames with information about the 2009 Kenya Population and Housing Census and the population in Kenya at the time of the 2014 Demographic Health Survey. Some of the data.frames have been adjusted to contain information about neonatals born from 2010-2014

rather than general population in 2014. The dataset names are: easpaKenya, easpaKenyaNeonatal, poppaKenya, and poppsubKenya.

An object of class `data.frame` with 47 rows and 12 columns.

An object of class `data.frame` with 47 rows and 6 columns.

An object of class `data.frame` with 300 rows and 7 columns.

Source

https://dhsprogram.com/pubs/pdf/FR308/FR308.pdf

References

Kenya National Bureau of Statistics, Ministry of Health/Kenya, National AIDS Control Council/Kenya, Kenya Medical Research Institute, and National Council For Population And Development/Kenya, 2015. Kenya Demographic and Health Survey 2014. Rockville, Maryland, USA. URL: http://dhsprogram.com/pubs/pdf/FR308/FR308.pdf.

Stevens, F.R., Gaughan, A.E., Linard, C., Tatem, A.J., 2015. Disaggregat- ing census data for population mapping using random forests with remotely-sensed and ancillary data. PloS One 10, e0107042.

Tatem, A.J., 2017. WorldPop, open data for spatial demography. Scientific Data 4.

`KingCounty` *Map of King County*

Description

Shapefiles are King County in the Washington States.

Usage

```
KingCounty
```

Format

An object of class `SpatialPolygonsDataFrame` with 48 rows and 9 columns.

logit *Logit transformation*

Description

Logit transformation

Usage

```
logit(x)
```

Arguments

x	data

Value

logit of x

Examples

```
x <- .5
logit(x)
```

logitNormMean *Calculate the mean of a distribution whose logit is Gaussian*

Description

Adapted from logitnorm package. Calculates the mean of a distribution whose logit is Gaussian. Each row of muSigmaMat is a mean and standard deviation on the logit scale.

Usage

```
logitNormMean(muSigmaMat, logisticApprox = FALSE, ...)
```

Arguments

muSigmaMat	An n x 2 matrix where each row is μ and σ on the logit scale for an independent random variable.
logisticApprox	Whether or not to use logistic approximation to speed up computation. See details for more information.
...	More arguments, passed to integrate function

Details

If logit$(Y) \sim N(\mu, \sigma^2)$, This function calculates $E[Y]$ via either numerical integration or by assuming that Y follows a logistic distribution. Under this approximation, setting $k = 16\sqrt{3}/(15\pi)$, we approximate the expectation as:

$$E[Y] = expit(\mu/\sqrt{1 + k^2\sigma^2})$$

The above logistic approximation speeds up the computation, but also sacrifices some accuracy.

Value

A vector of expectations of the specified random variables

Author(s)

John Paige

Examples

```
mus = c(-5, 0, 5)
sigmas = rep(1, 3)
logitNormMean(cbind(mus, sigmas))
logitNormMean(cbind(mus, sigmas), TRUE)
```

---

makePopIntegrationTab *Generating pixellated populations, and population frames*

Description

[Experimental]

Usage

```
makePopIntegrationTab(
  km.res = 5,
  pop,
  domain.map.dat,
  east.lim,
  north.lim,
  map.projection,
  area.map.dat,
  subarea.map.dat,
  areaNameVar = "NAME_1",
  subareaNameVar = "NAME_2",
  poppa = NULL,
  poppsub = NULL,
  stratify.by.urban = TRUE,
```

```
  areapa = NULL,
  areapsub = NULL,
  custom.subset.polygons = NULL,
  area.polygon.subset.I = NULL,
  subarea.polygon.subset.I = NULL,
  mean.neighbor = 50,
  delta = 0.1,
  return.popp.tables = FALSE,
  set.na.to.zero = TRUE,
  fix.zero.pop.density.subareas = FALSE,
  extract.method = "bilinear"
)

getPoppsub(
  km.res = 1,
  pop,
  domain.map.dat,
  east.lim,
  north.lim,
  map.projection,
  poppa,
  areapa = NULL,
  areapsub,
  subarea.map.dat,
  subareaNameVar = "NAME_2",
  stratify.by.urban = TRUE,
  area.map.dat = NULL,
  areaNameVar = "NAME_1",
  area.polygon.subset.I = NULL,
  subarea.polygon.subset.I = NULL,
  custom.subset.polygons = NULL,
  mean.neighbor = 50,
  delta = 0.1,
  set.na.to.zero = TRUE,
  fix.zero.pop.density.subareas = FALSE
)

adjustPopMat(
  pop.mat,
  poppa.target = NULL,
  adjust.by = c("area", "subarea"),
  stratify.by.urban = TRUE
)
```

Arguments

km.res	The resolution of the pixelated grid in km
pop	Population density raster

`domain.map.dat` A shapefile representing the full spatial domain (e.g. country)

`east.lim` Range in km easting over the spatial domain under the input projection

`north.lim` Range in km northing over the spatial domain under the input projection

`map.projection` A projection function taking longitude and latitude and returning easting and northing in km. Or the inverse if inverse is set to TRUE. For example, `projKenya`. Check https://epsg.io/ for example for best projection EPSG codes for specific countries

`area.map.dat` SpatialPolygonsDataFrame object with area level map information

`subarea.map.dat`
SpatialPolygonsDataFrame object with subarea level map information

`areaNameVar` The name of the area variable associated with `area.map.dat@data` and `subarea.map.dat@data`

`subareaNameVar` The name of the subarea variable associated with `subarea.map.dat@data`

`poppa` data.frame of population per area separated by urban/rural. If `poppsub` is not included, this is used for normalization of populations associated with population integration points. Contains variables:

area name of area

popUrb total urban (general) population of area

popRur total rural (general) population of area

popTotal total (general) population of area

pctUrb percentage of population in the area that is urban (between 0 and 100)

`poppsub` data.frame of population per subarea separated by urban/rural using for population normalization or urbanicity classification. Often based on extra fine scale population density grid. Has variables:

subarea name of subarea

area name of area

popUrb total urban (general) population of subarea

popRur total rural (general) population of subarea

popTotal total (general) population of subarea

pctUrb percentage of population in the subarea that is urban (between 0 and 100)

`stratify.by.urban`
Whether to stratify the pixellated grid by urban/rural. If TRUE, renormalizes population densities within areas or subareas crossed with urban/rural

`areapa` A list with variables:

area name of area

spatialArea spatial area of the subarea (e.g. in km^2)

`areapsub` A list with variables:

subarea name of subarea

spatialArea spatial area of the subarea (e.g. in km^2)

`custom.subset.polygons`
'SpatialPolygonsDataFrame' or 'SpatialPolygons' object to subset the grid over. This option can help reduce computation time relative to constructing the whole

grid and subsetting afterwards. area.polygon.subset.I or subarea.polygon.subset.I can be used when subsetting by areas or subareas in area.map.dat or subarea.map.dat. Must be in latitude/longitude projection "EPSG:4326"

area.polygon.subset.I
: Index in area.map.dat for a specific area to subset the grid over. This option can help reduce computation time relative to constructing the whole grid and subsetting afterwards

subarea.polygon.subset.I
: FOR EXPERIMENTAL PURPOSES ONLY. Index in subarea.map.dat for a specific area to subset the grid over. This option can help reduce computation time relative to constructing the whole grid and subsetting afterwards

mean.neighbor
: For determining what area or subarea points are nearest to if they do not directly fall into an area. See fields.rdist.near for details.

delta
: For determining what area or subarea points are nearest to if they do not directly fall into an area. See fields.rdist.near for details.

return.popp.tables
: If TRUE, poppa and poppsub will be calculated based on the generated population integration matrix and input area/subarea map data

set.na.to.zero
: If TRUE, sets NA populations to 0.

fix.zero.pop.density.subareas
: If TRUE, if population density in a subarea is estimated to be zero, but the total population in the subarea is nonzero, population is filled into the area uniformly

extract.method
: Either 'bilinear' or 'simple'. see method from extract

pop.mat
: Pixellated grid data frame with variables area and pop such as that generated by makePopIntegrationTab

poppa.target
: Target population per area stratified by urban rural. Same format as poppa

adjust.by
: Whether to adjust population density by the area or subarea level

Details

Functions for generating pixellated population information and population frames at the area and subarea levels. The area and subarea levels can be thought of as big regions and little regions, where areas can be partitioned into unique sets of subareas. For example, Admin-1 and Admin-2 areas might be areas and subareas respectively. The population totals are either tabulated at the area x urban/rural level, the subarea x urban/rural level, or at the pixel level of a specified resolution. Totals are calculated using population density information, shapefiles, and, possibly, preexisting population frames at different areal levels. Note that area names should each be unique, and similarly for subarea names.

Functions

- makePopIntegrationTab(): Generate pixellated grid of coordinates (both longitude/latitude and east/north) over spatial domain of the given resolution with associated population totals, areas, subareas, and urban/rural levels. For very small areas that might not otherwise have a grid point in them, a custom integration point is added at their centroid. Sets urbanicity classifications by thresholding input population density raster using area and subarea population

tables, and generates area and subarea population tables from population density information if not already given. Can be used for integrating predictions from the given coordinates to area and subarea levels using population weights.

- `getPoppsub()`: Generate table of estimates of population totals per subarea x urban/rural combination based on population density raster at `kmres` resolution "grid", including custom integration points for any subarea too small to include grid points at their centroids.
- `adjustPopMat()`: Adjust population densities in grid based on a population frame.

Author(s)

John Paige

See Also

`setThresholdsByRegion`, `poppRegionFromPopMat`, `simPopSPDE`, `simPopCustom`

Examples

```
## Not run:

library(sp)
library(sf)
# download Kenya GADM shapefiles from SUMMERdata github repository
githubURL <- paste0("https://github.com/paigejo/SUMMERdata/blob/main/data/",
                    "kenyaMaps.rda?raw=true")
tempDirectory = "~/"
mapsFilename = paste0(tempDirectory, "/kenyaMaps.rda")
if(!file.exists(mapsFilename)) {
  download.file(githubURL,mapsFilename)
}

# load it in
out = load(mapsFilename)
out
adm1@data$NAME_1 = as.character(adm1@data$NAME_1)
adm1@data$NAME_1[adm1@data$NAME_1 == "Trans Nzoia"] = "Trans-Nzoia"
adm1@data$NAME_1[adm1@data$NAME_1 == "Elgeyo-Marakwet"] = "Elgeyo Marakwet"
adm2@data$NAME_1 = as.character(adm2@data$NAME_1)
adm2@data$NAME_1[adm2@data$NAME_1 == "Trans Nzoia"] = "Trans-Nzoia"
adm2@data$NAME_1[adm2@data$NAME_1 == "Elgeyo-Marakwet"] = "Elgeyo Marakwet"

# some Admin-2 areas have the same name
adm2@data$NAME_2 = as.character(adm2@data$NAME_2)
adm2@data$NAME_2[(adm2@data$NAME_1 == "Bungoma") &
                   (adm2@data$NAME_2 == "Lugari")] = "Lugari, Bungoma"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Kakamega") &
                   (adm2@data$NAME_2 == "Lugari")] = "Lugari, Kakamega"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Meru") &
                   (adm2@data$NAME_2 == "Igembe South")] = "Igembe South, Meru"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Tharaka-Nithi") &
                   (adm2@data$NAME_2 == "Igembe South")] = "Igembe South, Tharaka-Nithi"
```

```
# The spatial area of unknown 8 is so small, it causes problems unless its removed or
# unioned with another subarea. Union it with neighboring Kakeguria:
newadm2 = adm2
unknown8I = which(newadm2$NAME_2 == "unknown 8")
newadm2$NAME_2[newadm2$NAME_2 %in% c("unknown 8", "Kapenguria")] <-
  "Kapenguria + unknown 8"
admin2.IDs <- newadm2$NAME_2

newadm2@data = cbind(newadm2@data, NAME_2OLD = newadm2@data$NAME_2)
newadm2@data$NAME_2OLD = newadm2@data$NAME_2
newadm2@data$NAME_2 = admin2.IDs
newadm2$NAME_2 = admin2.IDs
temp <- terra::aggregate(as(newadm2, "SpatVector"), by="NAME_2")

temp <- sf::st_as_sf(temp)
temp <- sf::as_Spatial(temp)

tempData = newadm2@data[-unknown8I,]
tempData = tempData[order(tempData$NAME_2),]
newadm2 <- sp::SpatialPolygonsDataFrame(temp, tempData, match.ID = F)
adm2 = newadm2

# download 2014 Kenya population density TIF file

githubURL <- paste0("https://github.com/paigejo/SUMMERdata/blob/main/data/",
                    "Kenya2014Pop/worldpop_total_1y_2014_00_00.tif?raw=true")
popTIFFilename = paste0(tempDirectory, "/worldpop_total_1y_2014_00_00.tif")
if(!file.exists(popTIFFilename)) {
  download.file(githubURL,popTIFFilename)
}

# load it in
pop = terra::rast(popTIFFilename)

east.lim = c(-110.6405, 832.4544)
north.lim = c(-555.1739, 608.7130)

## Construct poppsubKenya, a table of urban/rural general population totals
## in each subarea. Technically, this is not necessary since we can load in
## poppsubKenya via data(kenyaPopulationData). First, we will need to calculate
## the areas in km^2 of the areas and subareas


# use Lambert equal area projection of areas (Admin-1) and subareas (Admin-2)
midLon = mean(adm1@bbox[1,])
midLat = mean(adm1@bbox[2,])
p4s = paste0("+proj=laea +x_0=0 +y_0=0 +lon_0=", midLon,
             " +lat_0=", midLat, " +units=km")

adm1_sf = st_as_sf(adm1)
adm1proj_sf = st_transform(adm1_sf, p4s)
adm1proj = as(adm1proj_sf, "Spatial")
```

```
adm2_sf = st_as_sf(adm2)
adm2proj_sf = st_transform(adm2_sf, p4s)
adm2proj = as(adm2proj_sf, "Spatial")

# now calculate spatial area in km^2
admin1Areas = as.numeric(st_area(adm1proj_sf))
admin2Areas = as.numeric(st_area(adm2proj_sf))

areapaKenya = data.frame(area=adm1proj@data$NAME_1, spatialArea=admin1Areas)
areapsubKenya = data.frame(area=adm2proj@data$NAME_1, subarea=adm2proj@data$NAME_2,
                           spatialArea=admin2Areas)

# Calculate general population totals at the subarea (Admin-2) x urban/rural
# level and using 1km resolution population grid. Assign urbanicity by
# thresholding population density based on estimated proportion population
# urban/rural, making sure total area (Admin-1) urban/rural populations in
# each area matches poppaKenya.
require(fields)
# NOTE: the following function will typically take ~15-20 minutes. Can speed up
#       by setting km.res to be higher, but we recommend fine resolution for
#       this step, since it only needs to be done once. Instead of running
#       the code in the following if(FALSE) section,
#       you can simply run data(kenyaPopulationData)
if(FALSE){
  system.time(poppsubKenya <- getPoppsub(
    km.res=1, pop=pop, domain.map.dat=adm0,
    east.lim=east.lim, north.lim=north.lim, map.projection=projKenya,
    poppa = poppaKenya, areapa=areapaKenya, areapsub=areapsubKenya,
    area.map.dat=adm1, subarea.map.dat=adm2,
    areaNameVar = "NAME_1", subareaNameVar="NAME_2"))
}
data(kenyaPopulationData)

# Now generate a general population integration table at 5km resolution,
# based on subarea (Admin-2) x urban/rural population totals. This takes
# ~1 minute
system.time(pop.matKenya <- makePopIntegrationTab(
  km.res=5, pop=pop, domain.map.dat=adm0,
  east.lim=east.lim, north.lim=north.lim, map.projection=projKenya,
  poppa = poppaKenya, poppsub=poppsubKenya,
  area.map.dat = adm1, subarea.map.dat = adm2,
  areaNameVar = "NAME_1", subareaNameVar="NAME_2"))

## Adjust pop.mat to be target (neonatal) rather than general population density. First
## create the target population frame
## (these numbers are based on IPUMS microcensus data)
mothersPerHouseholdUrb = 0.3497151
childrenPerMotherUrb = 1.295917
mothersPerHouseholdRur = 0.4787696
childrenPerMotherRur = 1.455222
targetPopPerStratumUrban = easpaKenya$HHUrb * mothersPerHouseholdUrb * childrenPerMotherUrb
targetPopPerStratumRural = easpaKenya$HHRur * mothersPerHouseholdRur * childrenPerMotherRur
```

```
easpaKenyaNeonatal = easpaKenya
easpaKenyaNeonatal$popUrb = targetPopPerStratumUrban
easpaKenyaNeonatal$popRur = targetPopPerStratumRural
easpaKenyaNeonatal$popTotal = easpaKenyaNeonatal$popUrb + easpaKenyaNeonatal$popRur
easpaKenyaNeonatal$pctUrb = 100 * easpaKenyaNeonatal$popUrb / easpaKenyaNeonatal$popTotal
easpaKenyaNeonatal$pctTotal =
  100 * easpaKenyaNeonatal$popTotal / sum(easpaKenyaNeonatal$popTotal)

# Generate the target population density by scaling the current population density grid
# at the Admin1 x urban/rural level
pop.matKenyaNeonatal = adjustPopMat(pop.matKenya, easpaKenyaNeonatal)

# Generate neonatal population table from the neonatal population integration matrix.
# This is technically not necessary for population simulation purposes, but is here
# for illustrative purposes
poppsubKenyaNeonatal = poppRegionFromPopMat(pop.matKenyaNeonatal, pop.matKenyaNeonatal$subarea)
poppsubKenyaNeonatal = cbind(subarea=poppsubKenyaNeonatal$region,
                             area=adm2@data$NAME_1[match(poppsubKenyaNeonatal$region,
                               adm2@data$NAME_2)],
                             poppsubKenyaNeonatal[,-1])
print(head(poppsubKenyaNeonatal))

## End(Not run)
```

`MalawiData` *Auxiliary data for Malawi 2000, 2004, 2010, and 2015 DHS.*

Description

The list contains several data frames.

Usage

```
data(MalawiData)
```

Format

An object of class `list` of length 4.

Details

- HIV, a data frame with three columns: years (in five year periods), survey, and the estimated bias of the reported U5MR due to HIV for each 5 year period. The bias is represented as the ratio of the reported U5MR to the true U5MR.
- HIV.yearly, a data frame with three columns: years (in one year interval), survey, and the estimated bias of the reported U5MR due to HIV for each year. The bias is represented as the ratio of the reported U5MR to the true U5MR.
- IGME2019. Yearly Estimates of national under-5 child mortality in Malawi from the 2019 UN-IGME estimates.

- IGME2019.nmr. Yearly Estimates of national neonatal mortality in Malawi from the 2019 UN-IGME estimates.

References

Neff Walker, Kenneth Hill, and Fengmin Zhao (2012) *Child mortality estimation: methods used to adjust for bias due to aids in estimating trends in under-five mortality.*,
PLoS Medicine, 9(8):e1001298.

`MalawiMap` *Malawi Admin-2 map*

Description

SpatialPolygonsDataFrame objects that reflect the Admin 2 regions in Malawi, including the Likoma island. The Admin 2 region names are in the ADM2_EN field.

Usage

```
data(MalawiMap)
```

Format

An object of class `SpatialPolygonsDataFrame` with 28 rows and 14 columns.

`mapEstimates` *Mapping estimates for svysae object*

Description

Mapping estimates for svysae object

Usage

```
mapEstimates(x, geo.data, variable, viridis.option = "viridis")
```

Arguments

`x`	syvsae object
`geo.data`	sf object containing polygon data for the small areas. One of the columns should be named domain and contain the domain labels.
`variable`	The posterior summary variable to plot. May be one of "median", "mean", or "var".
`viridis.option`	viridis color scheme

Value

ggplot containing map of small area posterior summary statistics

Examples

```
## Not run:
data(DemoData2)
data(DemoMap2)
library(survey)
des0 <- svydesign(ids = ~clustid+id, strata = ~strata,
                  weights = ~weights, data = DemoData2, nest = TRUE)
Xmat <- aggregate(age~region, data = DemoData2, FUN = mean)
geo.data <- sf::st_as_sf(DemoMap2$geo)
geo.data$domain <- geo.data$REGNAME
cts.res <- smoothArea(tobacco.use ~ 1,
                      domain = ~region,
                      design = des0,
                      adj.mat = DemoMap2$Amat,
                      pc.u = 1,
                      pc.alpha = 0.01,
                      pc.u.phi = 0.5,
                      pc.alpha.phi = 2/3,
                      return.samples = TRUE)
mapEstimates(cts.res, geo.data = geo.data, variable = "median")
mapEstimates(cts.res, geo.data = geo.data, variable = "var")

## End(Not run)
```

`mapPlot` *Plot region-level variables on a map*

Description

This function visualizes the map with different variables. The input data frame can be either the long or wide format.

Usage

```
mapPlot(
  data = NULL,
  variables,
  values = NULL,
  labels = NULL,
  geo,
  by.data,
  by.geo,
  is.long = FALSE,
  size = 0.5,
```

```
  removetab = FALSE,
  border = "gray20",
  ncol = NULL,
  ylim = NULL,
  legend.label = NULL,
  per1000 = FALSE,
  clean = TRUE,
  size.label = 2,
  add.adj = FALSE,
  color.adj = "red",
  alpha.adj = 0.85,
  direction = 1,
  cut = NULL
)
```

Arguments

data	a data frame with variables to be plotted. When it is null, a map is produced.
variables	vector of variables to be plotted. If long format of data is used, only one variable can be selected
values	the column corresponding to the values to be plotted, only used when long format of data is used
labels	vector of labels to use for each variable, only used when wide format of data is used
geo	SpatialPolygonsDataFrame object for the map
by.data	column name specifying region names in the data
by.geo	variable name specifying region names in the data
is.long	logical indicator of whether the data is in the long format, default to FALSE
size	size of the border
removetab	logical indicator to not show the tab label, only applicable when only one tab is present.
border	color of the border
ncol	number of columns for the output tabs
ylim	range of the values to be plotted.
legend.label	Label for the color legend.
per1000	logical indicator to plot mortality rates as rates per 1,000 live births. Note that the added comparison data should always be in the probability scale.
clean	remove all coordinates for a cleaner layout, default to TRUE.
size.label	size of the label of the regions.
add.adj	logical indicator to add edges between connected regions.
color.adj	color of the adjacency matrix edges.
alpha.adj	alpha level (transparency) of the adjacency matrix edges.
direction	Direction of the color scheme. It can be either 1 (smaller values are darker) or -1 (higher values are darker). Default is set to 1.
cut	a vector of values to cut the continuous scale color to discrete intervals.

Author(s)

Zehang Richard Li

Examples

```
## Not run:
data(DemoMap)
# Plotting data in the long format
dat <- data.frame(region = rep(c("central",  "eastern", "northern", "western"), 3),
year = rep(c(1980, 1990, 2000), each = 4),
values = stats::rnorm(12))
utils::head(dat)
mapPlot(dat, variables = "year", values = "values",
by.data = "region", geo = DemoMap$geo,
by.geo = "NAME_final", is.long = TRUE)
dat <- data.frame(region = c("central",  "eastern", "northern", "western"),
Year1 = stats::rnorm(4), Year2 = stats::rnorm(4),
Year3 = stats::rnorm(4))
utils::head(dat)
mapPlot(dat, variables = c("Year1", "Year2", "Year3"),
 labels = c(1980, 1990, 2000),
by.data = "region", geo = DemoMap$geo,
by.geo = "NAME_final", is.long = FALSE)


## End(Not run)
```

---

mapPoints *Map GPS points to polygon regions*

Description

Map GPS points to polygon regions

Usage

```
mapPoints(data, geo, long, lat, names)
```

Arguments

data	point data with two columns of GPS locations.
geo	SpatialPolygonsDataFrame of the map
long	column name for longitudinal coordinate in the data
lat	column name for latitude coordinate in the data
names	character vector of region ids to be added to the neighbours list

Value

Spatial djacency matrix.

Author(s)

Zehang Richard Li

Examples

```
data(DemoMap)
dat <- data.frame(ID = c(1,2,3), lon = c(32.2, 33.7, 33), lat = c(0.1, 0.9, 2.8))
dat2 <- mapPoints(dat, DemoMap$geo, long = "lon", lat = "lat", names = "REGNAME")
dat2
```

---

plot.SUMMERproj *Plot projection output.*

Description

Plot projection output.

Usage

```
## S3 method for class 'SUMMERproj'
plot(
  x,
  year.label = c("85-89", "90-94", "95-99", "00-04", "05-09", "10-14", "15-19"),
  year_label = deprecated(),
  year.med = c(1987, 1992, 1997, 2002, 2007, 2012, 2017),
  year_med = deprecated(),
  is.subnational = TRUE,
  year.proj = 2015,
  proj_year = deprecated(),
  data.add = NULL,
  option.add = list(point = NULL, lower = NULL, upper = NULL, by = NULL),
  color.add = "black",
  label.add = NULL,
  dodge.width = 0.5,
  plot.CI = NULL,
  per1000 = FALSE,
  color.CI = NULL,
  alpha.CI = 0.5,
  ...
)
```

Arguments

x	output from getSmoothed
year.label	labels for the periods
year_label	**[Deprecated]** replaced by year.label
year.med	labels for the middle years in each period, only used when both yearly and period estimates are plotted. In that case, year.med specifies where each period estimates are aligned.
year_med	**[Deprecated]** replaced by year.med
is.subnational	logical indicator of whether the data contains subnational estimates
year.proj	the first year where projections are made, i.e., where no data are available.
proj_year	**[Deprecated]** replaced by year.proj
data.add	data frame for the Comparisons data points to add to the graph. This can be, for example, the raw direct estimates. This data frame is merged to the projections by column 'region' and 'years'. Except for these two columns, this dataset should not have Comparisons columns with names overlapping the getSmoothed output.
option.add	list of options specifying the variable names for the points to plot, lower and upper bounds, and the grouping variable. This is intended to be used to add Comparisons estimates on the same plot as the smoothed estimates. See examples for details.
color.add	the color of the Comparisons data points to plot.
label.add	the label of the Comparisons data points in the legend.
dodge.width	the amount to add to data points at the same year to avoid overlap. Default to be 0.5.
plot.CI	logical indicator of whether to plot the error bars.
per1000	logical indicator to plot mortality rates as rates per 1,000 live births. Note that the added comparison data should always be in the probability scale.
color.CI	the color of the error bars of the credible interval.
alpha.CI	the alpha (transparency) of the error bars of the credible interval.
...	optional arguments, see details

Author(s)

Zehang Richard Li

See Also

getSmoothed

Examples

```
## Not run:
years <- levels(DemoData[[1]]$time)

# obtain direct estimates
data <- getDirectList(births = DemoData,
years = years,
regionVar = "region", timeVar = "time",
clusterVar = "~clustid+id",
ageVar = "age", weightsVar = "weights",
geo.recode = NULL)
# obtain direct estimates
data_multi <- getDirectList(births = DemoData, years = years,
  regionVar = "region",  timeVar = "time", clusterVar = "~clustid+id",
  ageVar = "age", weightsVar = "weights", geo.recode = NULL)
data <- aggregateSurvey(data_multi)

#  national model
years.all <- c(years, "15-19")
fit1 <- smoothDirect(data = data, geo = NULL, Amat = NULL,
  year.label = years.all, year.range = c(1985, 2019),
  rw = 2, is.yearly=FALSE, m = 5)
out1 <- getSmoothed(fit1)
plot(out1, is.subnational=FALSE)

#  subnational model
fit2 <- smoothDirect(data = data, geo = geo, Amat = mat,
  year.label = years.all, year.range = c(1985, 2019),
  rw = 2, is.yearly=TRUE, m = 5, type.st = 4)
out2 <- getSmoothed(fit2)
plot(out2, is.yearly=TRUE, is.subnational=TRUE)



## End(Not run)
```

`poppRegionFromPopMat` *Generate a population frame of a similar format to poppa argument of* `simPopCustom` *with a custom set of regions*

Description

[Experimental]

Usage

```
poppRegionFromPopMat(pop.mat, regions)
```

Arguments

pop.mat	Pixellated grid data frame with variables area and pop. Assumed to be stratified by urban/rural
regions	character vector of length nPixels giving a custom set of regions for which to generate a population frame using population density

Details

Urbanicity thresholds are set based on that region's percent population urban. Intended as a helper function of getPoppsub, but can also be used for custom sets of regions (i.e. more than just 2 areal levels: area and subarea).

Value

A table of population totals by region

Author(s)

John Paige

See Also

getPoppsub

Examples

```
## Not run:
data(kenyaPopulationData)

#' # download Kenya GADM shapefiles from SUMMERdata github repository
githubURL <- "https://github.com/paigejo/SUMMERdata/blob/main/data/kenyaMaps.rda?raw=true"
tempDirectory = "~/"
mapsFilename = paste0(tempDirectory, "/kenyaMaps.rda")
if(!file.exists(mapsFilename)) {
  download.file(githubURL,mapsFilename)
}

# load it in
out = load(mapsFilename)
out
kenyaMesh <- fmesher::fm_as_fm(kenyaMesh)
adm1@data$NAME_1 = as.character(adm1@data$NAME_1)
adm1@data$NAME_1[adm1@data$NAME_1 == "Trans Nzoia"] = "Trans-Nzoia"
adm1@data$NAME_1[adm1@data$NAME_1 == "Elgeyo-Marakwet"] = "Elgeyo Marakwet"
adm2@data$NAME_1 = as.character(adm2@data$NAME_1)
adm2@data$NAME_1[adm2@data$NAME_1 == "Trans Nzoia"] = "Trans-Nzoia"
adm2@data$NAME_1[adm2@data$NAME_1 == "Elgeyo-Marakwet"] = "Elgeyo Marakwet"

# some Admin-2 areas have the same name
adm2@data$NAME_2 = as.character(adm2@data$NAME_2)
adm2@data$NAME_2[(adm2@data$NAME_1 == "Bungoma") &
```

```
  (adm2@data$NAME_2 == "Lugari")] = "Lugari, Bungoma"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Kakamega") &
  (adm2@data$NAME_2 == "Lugari")] = "Lugari, Kakamega"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Meru") &
  (adm2@data$NAME_2 == "Igembe South")] = "Igembe South, Meru"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Tharaka-Nithi") &
  (adm2@data$NAME_2 == "Igembe South")] = "Igembe South, Tharaka-Nithi"

# The spatial area of unknown 8 is so small, it causes problems unless
# its removed or unioned with another subarea. Union it with neighboring
# Kakeguria:
newadm2 = adm2
unknown8I = which(newadm2$NAME_2 == "unknown 8")
newadm2$NAME_2[newadm2$NAME_2 %in% c("unknown 8", "Kapenguria")] <- "Kapenguria + unknown 8"
admin2.IDs <- newadm2$NAME_2

newadm2@data = cbind(newadm2@data, NAME_2OLD = newadm2@data$NAME_2)
newadm2@data$NAME_2OLD = newadm2@data$NAME_2
newadm2@data$NAME_2 = admin2.IDs
newadm2$NAME_2 = admin2.IDs
temp <- terra::aggregate(as(newadm2, "SpatVector"), by="NAME_2")

library(sf)
temp <- sf::st_as_sf(temp)
temp <- sf::as_Spatial(temp)

tempData = newadm2@data[-unknown8I,]
tempData = tempData[order(tempData$NAME_2),]
newadm2 <- SpatialPolygonsDataFrame(temp, tempData, match.ID = F)
adm2 = newadm2

# download 2014 Kenya population density TIF file

githubURL <- paste0("https://github.com/paigejo/SUMMERdata/blob/main/data/",
                    "Kenya2014Pop/worldpop_total_1y_2014_00_00.tif?raw=true")
popTIFFilename = paste0(tempDirectory, "/worldpop_total_1y_2014_00_00.tif")
if(!file.exists(popTIFFilename)) {
  download.file(githubURL,popTIFFilename)
}

# load it in
pop = terra::rast(popTIFFilename)

east.lim = c(-110.6405, 832.4544)
north.lim = c(-555.1739, 608.7130)

require(fields)
#'
# Now generate a general population integration table at 5km resolution,
# based on subarea (Admin-2) x urban/rural population totals. This takes
# ~1 minute
pop.matKenya <- makePopIntegrationTab(
  km.res=5, pop=pop, domain.map.dat=adm0,
```

```
  east.lim=east.lim, north.lim=north.lim, map.projection=projKenya,
  poppa = poppaKenya, poppsub=poppsubKenya,
  area.map.dat = adm1, subarea.map.dat = adm2,
  areaNameVar = "NAME_1", subareaNameVar="NAME_2")

out = poppRegionFromPopMat(pop.matKenya, pop.matKenya$area)
out
poppaKenya

out = poppRegionFromPopMat(pop.matKenya, pop.matKenya$subarea)
out
poppsubKenya

pop.matKenyaUnstratified = pop.matKenya
pop.matKenyaUnstratified$urban = NULL
out = poppRegionFromPopMat(pop.matKenyaUnstratified, pop.matKenyaUnstratified$area)
out
poppaKenya

## End(Not run)
```

`print.SUMMERmodel` *Print method for the smoothing models.*

Description

This function is the print method for class `SUMMERmodel`.

Usage

```
## S3 method for class 'SUMMERmodel'
print(x, ...)
```

Arguments

`x`	output from `smoothDirect` or `smoothCluster`
`...`	not used

Author(s)

Zehang Li

See Also

`summary.SUMMERmodel`

Examples

```
## Not run:
  library(SUMMER)
  library(dplyr)
  data(DemoData)

  # Smooth Direct Model
  years <- levels(DemoData[[1]]$time)
  # obtain direct estimates
  data_multi <- getDirectList(births = DemoData, years = years,
  regionVar = "region",  timeVar = "time", clusterVar = "~clustid+id",
  ageVar = "age", weightsVar = "weights", geo.recode = NULL)
  data <- aggregateSurvey(data_multi)

  years.all <- c(years, "15-19")
  fit <- smoothDirect(data = data, Amat = NULL,
  year.label = years.all, year.range = c(1985, 2019),
  time.model = 'rw2', is.yearly=FALSE, m = 5)
  fit

  # Cluster-level Model
  counts.all <- NULL
  for(i in 1:length(DemoData)){
  counts <- getCounts(DemoData[[i]][, c("clustid", "time", "age", "died",
                                        "region", "strata")],
           variables = 'died', by = c("age", "clustid", "region",
                                      "time", "strata"))
  counts <- counts %>% mutate(cluster = clustid, years = time, Y=died)
  counts$strata <- gsub(".*\\.","",counts$strata)
  counts$survey <- names(DemoData)[i]
  counts.all <- rbind(counts.all, counts)
  }

  # fit cluster-level model on the periods
  periods <- levels(DemoData[[1]]$time)
  fit <- smoothCluster(data = counts.all,
     Amat = DemoMap$Amat,
     time.model = "rw2",
     st.time.model = "rw1",
     strata.time.effect =  TRUE,
     survey.effect = TRUE,
     family = "betabinomial",
     year.label = c(periods, "15-19"))
  fit

## End(Not run)
```

---

print.SUMMERmodel.svy *Print method for the smoothing models from* smoothSurvey.

Description

This function is the print method for class SUMMERmodel.svy.

Usage

```
## S3 method for class 'SUMMERmodel.svy'
print(x, ...)
```

Arguments

x	output from smoothSurvey.
...	not used

Author(s)

Zehang Li

See Also

summary.SUMMERmodel.svy

Examples

```
## Not run:
data(DemoData2)
data(DemoMap2)
fit0 <- smoothSurvey(data=DemoData2,
Amat=DemoMap2$Amat, responseType="binary",
responseVar="tobacco.use", strataVar="strata",
weightVar="weights", regionVar="region",
clusterVar = "~clustid+id", CI = 0.95)
fit0

## End(Not run)
```

---

print.SUMMERprojlist *Print method for the combined projection output.*

Description

This function is the print method for class SUMMERprojlist.

Usage

```
## S3 method for class 'SUMMERprojlist'
print(x, ...)
```

Arguments

x	output from getSmoothed
...	not used

Author(s)

Zehang Li

Examples

```
## Not run:
 library(SUMMER)
 library(dplyr)
 data(DemoData)
 # Create dataset of counts
 counts.all <- NULL
 for(i in 1:length(DemoData)){
 counts <- getCounts(DemoData[[i]][, c("clustid", "time", "age", "died",
                                       "region", "strata")],
          variables = 'died', by = c("age", "clustid", "region",
                                        "time", "strata"))
 counts <- counts %>% mutate(cluster = clustid, years = time, Y=died)
 counts$strata <- gsub(".*\\.","",counts$strata)
 counts$survey <- names(DemoData)[i]
 counts.all <- rbind(counts.all, counts)
 }

 # fit cluster-level model on the periods
 periods <- levels(DemoData[[1]]$time)
 fit <- smoothCluster(data = counts.all,
    Amat = DemoMap$Amat,
    time.model = "rw2",
    st.time.model = "rw1",
    strata.time.effect =  TRUE,
    survey.effect = TRUE,
    family = "betabinomial",
    year.label = c(periods, "15-19"))
 summary(fit)
 est <- getSmoothed(fit, nsim = 1000)

## End(Not run)
```

---

projKenya *Map projection for Kenya*

Description

[Experimental]

Usage

```
projKenya(lon, lat = NULL, inverse = FALSE)
```

Arguments

lon	either longitude or, if inverse == TRUE, easting in km
lat	either latitude or, if inverse == TRUE, northing in km
inverse	if FALSE, projects from lon/lat to easting/northing. Else from easting/northing to lon/lat

Details

Projection specifically chosen for Kenya. Project from lat/lon to northing/easting in kilometers. Uses epsg=21097 with km units. May not work on all systems due to differences in the behavior between different PROJ and GDAL versions.

Value

A 2 column matrix of easting/northing coordinates in km if inverse == FALSE. Otherwise, a 2 column matrix of longitude/latitude coordinates.

Author(s)

John Paige

Examples

```
eastLim = c(-110.6405, 832.4544)
northLim = c(-555.1739, 608.7130)
coordMatrixEN = cbind(eastLim, northLim)
coordMatrixLL = projKenya(coordMatrixEN, inverse=TRUE)

coordMatrixLL
# if the coordMatrixLL isn't the following, projKenya may not support
# your installation of GDAL and/or PROJ:
#      east north
# [1,] 33.5  -5.0
# [2,] 42.0   5.5

projKenya(coordMatrixLL, inverse=FALSE)
# regardless of your PROJ/GDAL installations, the result of the
# above line of could should be:
#            lon       lat
# [1,] -110.6405 -555.1739
# [2,]  832.4544  608.7130
```

ridgePlot *Calculate and plot posterior densities of the projected estimates*

Description

The function ridgePlot replaces the previous function name getSmoothedDensity (before version 1.0.0).

Usage

```
ridgePlot(
  x = NULL,
  nsim = 1000,
  draws = NULL,
  year.plot = NULL,
  year_plot = deprecated(),
  strata.plot = NULL,
  strata_plot = deprecated(),
  by.year = TRUE,
  ncol = 4,
  scale = 2,
  per1000 = FALSE,
  order = 0,
  direction = 1,
  linewidth = 0.5,
  results = NULL,
  save.density = FALSE,
  ...
)
```

Arguments

x	output from smoothDirect for the smoothed direct estimates, or smoothCluster for the cluster-level estimates.
nsim	number of posterior draws to take. Only used for cluster-level models when draws is NULL. Otherwise the posterior draws in draws will be used instead without resampling.
draws	Output of getSmoothed with save.draws set to TRUE. This argument allows the previously sampled draws (by setting save.draws to be TRUE) be used in new aggregation tasks. This argument is only used for cluster-level models.
year.plot	A vector indicate which years to plot
year_plot	**[Deprecated]** replaced by year.plot
strata.plot	Name of the strata to plot. If not specified, the overall is plotted.
strata_plot	**[Deprecated]** replaced by strata.plot
by.year	logical indicator for whether the output uses years as facets.

ncol	number of columns in the output figure.
scale	numerical value controlling the height of the density plots.
per1000	logical indicator to multiply results by 1000.
order	order of regions when by.year is set to TRUE. Negative values indicate regions are ordered from high to low posterior medians from top to bottom. Positive values indicate from low to high. 0 indicate alphabetic orders.
direction	Direction of the color scheme. It can be either 1 (smaller values are darker) or -1 (higher values are darker). Default is set to 1.
linewidth	width of the ridgeline.
results	output from ridgePlot returned object with save.density = TRUE. This argument can be specified to avoid calculating densities again when only the visualization changes.
save.density	Logical indicator of whether the densities will be returned with the ggplot object. If set to TRUE, the output will be a list consisting of (1) a data frame of computed densities and (2) a ggplot object of the plot.
...	additional configurations passed to inla.posterior.sample.

Value

ridge plot of the density, and if save.density = TRUE, also a data frame of the calculated densities

Author(s)

Zehang Richard Li

See Also

plot.SUMMERproj

Examples

```
## Not run:
years <- levels(DemoData[[1]]$time)

data <- getDirectList(births = DemoData,
years = years,
regionVar = "region", timeVar = "time",
clusterVar = "~clustid+id",
ageVar = "age", weightsVar = "weights",
geo.recode = NULL)
# obtain direct estimates
data_multi <- getDirectList(births = DemoData, years = years,
  regionVar = "region",  timeVar = "time", clusterVar = "~clustid+id",
  ageVar = "age", weightsVar = "weights", geo.recode = NULL)
data <- aggregateSurvey(data_multi)

#  national model
years.all <- c(years, "15-19")
```

```
fit1 <- smoothDirect(data = data, geo = NULL, Amat = NULL,
  year.label = years.all, year.range = c(1985, 2019),
  rw = 2, m = 5)
## Plot marginal posterior densities over time
ridgePlot(fit1, year.plot = years.all,
          ncol = 4, by.year = FALSE)

#  subnational model
fit2 <- smoothDirect(data = data, geo = DemoMap$geo, Amat = DemoMap$Amat,
  year.label = years.all, year.range = c(1985, 2019),
  rw = 2, m = 5, type.st = 1)

# Plot marginal posterior densities over time (regions are ordered alphabetically)
ridgePlot(fit2, year.plot = years.all, ncol = 4)

# Re-order the regions and save the density to avoid re-compute later
density <- ridgePlot(fit2, year.plot = years.all,
 ncol = 4, per1000 = TRUE, order = -1, save.density = TRUE)
density$g

# Show each region (instead of each year) in a panel
## Instead of recalculate the posteriors, we can use previously calculated densities as input
ridgePlot(results = density, year.plot = years.all,
ncol = 4, by.year=FALSE, per1000 = TRUE)

# Show more years
ridgePlot(results = density, year.plot = c(1990:2019),
ncol = 4, by.year=FALSE, per1000 = TRUE)


# Example using surveyPrev package output

library(surveyPrev)
dhsData <- getDHSdata(country = "Rwanda", indicator = "nmr", year = 2019)
data <- getDHSindicator(dhsData, indicator = "nmr")
geo <- getDHSgeo(country = "Rwanda", year = 2019)
poly.adm1 <- geodata::gadm(country="RWA", level=1, path=tempdir())
poly.adm1 <- sf::st_as_sf(poly.adm1)
poly.adm2 <- geodata::gadm(country="RWA", level=2, path=tempdir())
poly.adm2 <- sf::st_as_sf(poly.adm2)
cluster.info <- clusterInfo(geo = geo,
              poly.adm1 = poly.adm1,
              poly.adm2 = poly.adm2,
                           by.adm1 = "NAME_1",
                           by.adm2 = "NAME_2")

fit1 <- directEST(data = data, cluster.info = cluster.info,  admin = 1)
fit2 <- directEST(data = data, cluster.info = cluster.info,  admin = 2)
ridgePlot(fit1, direction = -1)
ridgePlot(fit2, direction = -1)


## End(Not run)
```

rst *Simulate spatial and temporal random effects*

Description

This function simulates spatial and temporal random effects with mean zero. The method is described in Algorithm 3.1 of Rue & Held 2015.

Usage

```
rst(
  n = 1,
  type = c("s", "t", "st")[1],
  type.s = "ICAR",
  type.t = c("RW1", "RW2")[2],
  Amat = NULL,
  n.t = NULL,
  scale.model = TRUE
)
```

Arguments

n	sample size
type	type of random effects: temporal (t), spatial (s), or spatial-temporal (st)
type.s	type of spatial random effect, currently only ICAR is available
type.t	type of temporal random effect, currently only RW1 and RW2 are available
Amat	adjacency matrix for the spatial regions
n.t	number of time points for the temporal random effect
scale.model	logical indicator of whether to scale the random effects to have unit generalized variance. See Sørbye 2013 for more details

Value

a matrix (for spatial or temporal) or a three-dimensional array (for spatial-temporal) of the random effects.

Author(s)

Zehang Richard Li

References

Rue, H., & Held, L. (2005). *Gaussian Markov random fields: theory and applications*. CRC press.

Sørbye, S. H. (2013). *Tutorial: Scaling IGMRF-models in R-INLA*. Department of Mathematics and Statistics, University of Tromsø.

Examples

```
## Not run:
data(DemoMap)
## Spatial random effects
out <- rst(n=10000, type = "s", Amat = DemoMap$Amat)
# To verify the mean under the conditional specification
mean(out[,1] - apply(out[,c(2,3,4)], 1, mean))
mean(out[,2] - apply(out[,c(1,3)], 1, mean))
mean(out[,3] - apply(out[,c(1,2,4)], 1, mean))
mean(out[,4] - apply(out[,c(1,3)], 1, mean))

## Temporal random effects (RW1)
out <- rst(n=1, type = "t", type.t = "RW1", n.t = 200, scale.model = FALSE)
par(mfrow = c(1,2))
plot(1:dim(out)[2], out, col = 1, type = "l", xlab = "Time", ylab = "Random effects")
# verify the first order difference is normally distributed
first_diff <- diff(as.numeric(out[1,]))
qqnorm(first_diff )
abline(c(0,1))

## Temporal random effects (RW2)
out <- rst(n=1, type = "t", type.t = "RW2", n.t = 200, scale.model = FALSE)
par(mfrow = c(1,2))
plot(1:dim(out)[2], out, col = 1, type = "l", xlab = "Time", ylab = "Random effects")
# verify the second order difference is normally distributed
first_diff <- diff(as.numeric(out[1,]))
second_diff <- diff(first_diff)
qqnorm(second_diff)
abline(c(0,1))

## Spatial-temporal random effects
out <- rst(n=1, type = "st", type.t = "RW2", Amat = DemoMap$Amat, n.t = 50)
dimnames(out)
par(mfrow = c(1,1))
plot(1:dim(out)[3], out[1,1,], col = 1,
 type = "l", ylim = range(out), xlab = "Time", ylab = "Random effects")
for(i in 2:4) lines(1:dim(out)[3], out[1,i,], col = i)
legend("bottomright", colnames(DemoMap$Amat), col = c(1:4), lty = rep(1,4))

## End(Not run)
```

`rw.new` *New random walk 1 and 2 models for m-year to period random effects*

Description

New random walk 1 and 2 models for m-year to period random effects

Usage

```
rw.new(
  cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
  theta = NULL
)
```

Arguments

cmd	list of model components
theta	log precision

---

rw.new.pc *New random walk 1 and 2 models for m-year to period random effects*

Description

New random walk 1 and 2 models for m-year to period random effects

Usage

```
rw.new.pc(
  cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
  theta = NULL
)
```

Arguments

cmd	list of model components
theta	log precision

---

setThresholdsByRegion **[Experimental]**

Description

Set thresholds of population density for urbanicity classifications within each region of the given type

Usage

```
setThresholdsByRegion(pop.mat, poppr, region.type = "area")
```

Arguments

pop.mat	pixellated population density data frame with variables region.type and pop
poppr	A table with population totals by region of the given type (e.g. poppa or poppsub from makePopIntegrationTab)
region.type	The variable name from poppr giving the region names. Defaults to "area"

Details

Thresholds are set based on that region's percent population urban. Intended as a helper function of makePopIntegrationTab.

Value

A list of region names and their urbanicity thresholds in population density

Author(s)

John Paige

See Also

makePopIntegrationTab

Examples

```
## Not run:
data(kenyaPopulationData)

#' # download Kenya GADM shapefiles from SUMMERdata github repository
githubURL <- "https://github.com/paigejo/SUMMERdata/blob/main/data/kenyaMaps.rda?raw=true"
tempDirectory = "~/"
mapsFilename = paste0(tempDirectory, "/kenyaMaps.rda")
if(!file.exists(mapsFilename)) {
  download.file(githubURL,mapsFilename)
}

# load it in
out = load(mapsFilename)
out
kenyaMesh <- fmesher::fm_as_fm(kenyaMesh)
adm1@data$NAME_1 = as.character(adm1@data$NAME_1)
adm1@data$NAME_1[adm1@data$NAME_1 == "Trans Nzoia"] = "Trans-Nzoia"
adm1@data$NAME_1[adm1@data$NAME_1 == "Elgeyo-Marakwet"] = "Elgeyo Marakwet"
adm2@data$NAME_1 = as.character(adm2@data$NAME_1)
adm2@data$NAME_1[adm2@data$NAME_1 == "Trans Nzoia"] = "Trans-Nzoia"
adm2@data$NAME_1[adm2@data$NAME_1 == "Elgeyo-Marakwet"] = "Elgeyo Marakwet"

# some Admin-2 areas have the same name
adm2@data$NAME_2 = as.character(adm2@data$NAME_2)
adm2@data$NAME_2[(adm2@data$NAME_1 == "Bungoma") &
  (adm2@data$NAME_2 == "Lugari")] = "Lugari, Bungoma"
```

```
adm2@data$NAME_2[(adm2@data$NAME_1 == "Kakamega") &
  (adm2@data$NAME_2 == "Lugari")] = "Lugari, Kakamega"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Meru") &
  (adm2@data$NAME_2 == "Igembe South")] = "Igembe South, Meru"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Tharaka-Nithi") &
  (adm2@data$NAME_2 == "Igembe South")] = "Igembe South, Tharaka-Nithi"

# The spatial area of unknown 8 is so small, it causes problems unless
# its removed or unioned with another subarea. Union it with neighboring
# Kakeguria:
newadm2 = adm2
unknown8I = which(newadm2$NAME_2 == "unknown 8")
newadm2$NAME_2[newadm2$NAME_2 %in% c("unknown 8", "Kapenguria")] <- "Kapenguria + unknown 8"
admin2.IDs <- newadm2$NAME_2

newadm2@data = cbind(newadm2@data, NAME_2OLD = newadm2@data$NAME_2)
newadm2@data$NAME_2OLD = newadm2@data$NAME_2
newadm2@data$NAME_2 = admin2.IDs
newadm2$NAME_2 = admin2.IDs
temp <- terra::aggregate(as(newadm2, "SpatVector"), by="NAME_2")

library(sf)
temp <- sf::st_as_sf(temp)
temp <- sf::as_Spatial(temp)

tempData = newadm2@data[-unknown8I,]
tempData = tempData[order(tempData$NAME_2),]
newadm2 <- sp::SpatialPolygonsDataFrame(temp, tempData, match.ID = F)
adm2 = newadm2

# download 2014 Kenya population density TIF file

githubURL <- paste0("https://github.com/paigejo/SUMMERdata/blob/main/data/",
                    "Kenya2014Pop/worldpop_total_1y_2014_00_00.tif?raw=true")
popTIFFilename = paste0(tempDirectory, "/worldpop_total_1y_2014_00_00.tif")
if(!file.exists(popTIFFilename)) {
  download.file(githubURL,popTIFFilename)
}

# load it in
pop = terra::rast(popTIFFilename)

east.lim = c(-110.6405, 832.4544)
north.lim = c(-555.1739, 608.7130)

require(fields)

data(kenyaPopulationData)

# Now generate a general population integration table at 5km resolution,
# based on subarea (Admin-2) x urban/rural population totals. This takes
# ~1 minute
pop.matKenya <- makePopIntegrationTab(
```

```
  km.res=5, pop=pop, domain.map.dat=adm0,
  east.lim=east.lim, north.lim=north.lim, map.projection=projKenya,
  poppa = poppaKenya, poppsub=poppsubKenya,
  area.map.dat = adm1, subarea.map.dat = adm2,
  areaNameVar = "NAME_1", subareaNameVar="NAME_2")

out = setThresholdsByRegion(pop.matKenya, poppaKenya)
out

out = setThresholdsByRegion(pop.matKenya, poppsubKenya, region.type="subarea")
out

## End(Not run)
```

---

simhyper *Simulate hyperpriors from an GMRF*

Description

Simulate hyperpriors from an GMRF

Usage

```
simhyper(
  R = 2,
  nsamp = 1e+05,
  nsamp.check = 5000,
  Amat = NULL,
  nperiod = 6,
  only.iid = TRUE
)
```

Arguments

R	Desired prior odds ratio. Default to 2, i.e., a 95% prior interval for the residual odds ratios lies in the interval (R, 1/R).
nsamp	Sample to simulate for scaling factor
nsamp.check	Sample to simulate for checking range
Amat	Adjacency matrix of the areas in the data.
nperiod	numerical value of how many time periods in the data
only.iid	Indicator for whether or not only IID hyperpriors are simulated

Author(s)

Zehang Richard Li, Laina Mercer

References

Wakefield, J. Multi-level modelling, the ecologic fallacy, and hybrid study designs. *International Journal of Epidemiology*, 2009, vol. 38 (pg. 330-336).

Examples

```
## Not run:
data(DemoMap)
mat <- DemoMap$Amat
priors <- simhyper(R = 2, nsamp = 1e+05, nsamp.check = 5000, Amat = mat)

## End(Not run)
```

`simPop` *Simulate populations and areal prevalences*

Description

Given a spatial risk model, simulate populations and population prevalences at the enumeration area level (represented as points), and aggregate to the pixel and administrative areal level.

Usage

```
simPopSPDE(
  nsim = 1,
  easpa,
  pop.mat,
  target.pop.mat,
  poppsub,
  spde.mesh,
  marg.var = 0.243,
  sigma.epsilon = sqrt(0.463),
  gamma = 0.009,
  eff.range = 406.51,
  beta0 = -3.922,
  seed = NULL,
  inla.seed = -1L,
  n.HH.sampled = 25,
  stratify.by.urban = TRUE,
  subarea.level = TRUE,
  do.fine.scale.risk = FALSE,
  do.smooth.risk = FALSE,
  do.smooth.risk.logistic.approx = FALSE,
  min1.per.subarea = TRUE
)

simPopCustom(
```

```
  logit.risk.draws,
  sigma.epsilon.draws,
  easpa,
  pop.mat,
  target.pop.mat,
  stratify.by.urban = TRUE,
  validation.pixel.I = NULL,
  validation.cluster.I = NULL,
  clusters.per.pixel = NULL,
  do.fine.scale.risk = FALSE,
  do.smooth.risk = FALSE,
  do.smooth.risk.logistic.approx = FALSE,
  poppsub = NULL,
  subarea.level = FALSE,
  min1.per.subarea = TRUE,
  return.EA.info = FALSE,
  epsc = NULL
)
```

Arguments

nsim	Number of simulations
easpa	data.frame of enumeration area, households, and target population per area stratified by urban/rural with variables: **area** name of area **EAUrb** number of urban enumeration areas in the area **EARur** number of rural enumeration areas in the area **EATotal** total number of enumeration areas in the area **HHUrb** number of urban households in the area **HHRur** number of rural households in the area **HHTotal** total number of households in the area **popUrb** total urban (target) population of area **popRur** total rural (target) population of area **popTotal** total (general) population of area
pop.mat	Pixellated grid data frame with variables lon, lat, pop, area, subareas (if subarea.level is TRUE), urban (if stratify.by.urban is TRUE), east, and north
target.pop.mat	Same as pop.mat, but pop variable gives target rather than general population
poppsub	data.frame of population per subarea separated by urban/rural using for population density grid normalization or urbanicity classification. Often based on extra fine scale population density grid. Has variables:
spde.mesh	Triangular mesh for the SPDE
marg.var	Marginal variance of the spatial process, excluding cluster effects. If 0, no spatial component is included
sigma.epsilon	Standard deviation on the logit scale for iid Gaussian EA level random effects in the risk model

`gamma` Effect of urban on logit scale for logit model for risk

`eff.range` Effective spatial range for the SPDE model

`beta0` Intercept of logit model for risk

`seed` Random number generator seed

`inla.seed` Seed input to inla.qsample. 0L sets seed intelligently, > 0 sets a specific seed, < 0 keeps existing RNG

`n.HH.sampled` Number of households sampled per enumeration area. Default is 25 to match DHS surveys

`stratify.by.urban`
Whether or not to stratify simulations by urban/rural classification

`subarea.level` Whether or not to aggregate the population by subarea

`do.fine.scale.risk`
Whether or not to calculate the fine scale risk at each aggregation level in addition to the prevalence

`do.smooth.risk` Whether or not to calculate the smooth risk at each aggregation level in addition to the prevalence

`do.smooth.risk.logistic.approx`
Whether to use logistic approximation when calculating smooth risk. See `logitNormMean` for details.

`min1.per.subarea`
If TRUE, ensures there is at least 1 EA per subarea. If subareas are particularly unlikely to have enumeration areas since they have a very low proportion of the population in an area, then setting this to TRUE may be computationally intensive.

`logit.risk.draws`
nIntegrationPoints x nsim dimension matrix of draws from the pixel leve risk field on logit scale, leaving out potential nugget/cluster/EA level effects.

`sigma.epsilon.draws`
nsim length vector of draws of cluster effect logit scale SD (joint draws with logit.risk.draws)

`validation.pixel.I`
CURRENTLY FOR TESTING PURPOSES ONLY a set of indices of pixels for which we want to simulate populations (used for pixel level validation)

`validation.cluster.I`
CURRENTLY FOR TESTING PURPOSES ONLY a set of indices of cluster for which we want to simulate populations (used for cluster level validation)

`clusters.per.pixel`
CURRENTLY FOR TESTING PURPOSES ONLY Used for pixel level validation. Fixes the number of EAs per pixel.

`return.EA.info` If TRUE, returns information on every individual EA (BAU) for each simulated population

`epsc` nEAs x nsim matrix of simulated EA (BAU) level iid effects representing fine scale variation in risk. If NULL, they are simulated as iid Gaussian on a logit scale with SD given by sigma.epsilon.draws list(pixelPop=outPixelLevel, subareaPop=outSubareaLevel, areaPop=outAreaLevel, logit.risk.draws=logit.risk.draws)

Details

[Experimental]

For population simulation and aggregation, we consider three models: smooth risk, fine scale risk, and the fine scale prevalence. All will be described in detail in a paper in preparation. In the smooth risk model, pixel level risks are integrated with respect to target population density when producing areal estimates on a prespecified set of integration points. The target population may be, for example, neonatals rather than the general population. In the fine scale models, enumeration areas (EAs) are simulated as point locations and iid random effects in the EA level risk are allowed. EAs and populations are dispersed conditional on the (possibly approximately) known number of EAs, households, and target population at a particular areal level (these we call `areas`) using multilevel multinomial sampling, first sampling the EAs, then distributing households among the EAs, then the target population among the households. Any areal level below the `areas` we call `subareas`. For instance, the `areas` might be Admin-1 if that is the smallest level at which the number of EAs, households, and people is known, and the `subareas` might be Admin-2. The multilevel multinomial sampling may be stratified by urban/rural within the areas if the number of EAs, households, and people is also approximately known at that level.

Within each EA we assume a fixed probability of an event occurring, which is the fine scale `risk`. The fine scale `prevalence` is the empirical proportion of events within that EA. We assume EA level logit scale iid N(0, sigma.epsilon^2) random effects in the risk model. When averaged with equal weights over all EAs in an areal unit, this forms the fine scale risk. When instead the population numerators and denominators are aggregated, and are used to calculate the empirical proportion of events occurring in an areal unit, the resulting quantity is the fine scale prevalence in that areal unit.

Note that these functions can be used for either simulating populations for simulation studies, or for generating predictions accounting for uncertainty in EA locations and fine scale variation occurring at the EA level due to EA level iid random effects. Required, however, is a separately fit EA level spatial risk model and information on the spatial population density and the population frame.

Value

The simulated population aggregated to the enumeration area, pixel, subarea (generally Admin2), and area (generally Admin1) levels. Output includes:

`pixelPop`	A list of pixel level population aggregates
`subareaPop`	A list of `subarea` level population aggregates
`areaPop`	A list of `area` level population aggregates

Each of these contains population numerator and denominator as well as prevalence and risk information aggregated to the appropriate level.

Functions

- `simPopSPDE()`: Simulate populations and population prevalences given census frame and population density information. Uses SPDE model for generating spatial risk and can include iid cluster level effect.
- `simPopCustom()`: Simulate populations and population prevalences given census frame and population density information. Uses custom spatial logit risk function and can include iid cluster level effect.

Author(s)

John Paige

References

Paige, John, Geir-Arne Fuglstad, Andrea Riebler, and Jon Wakefield. "Spatial aggregation with respect to a population distribution: Impact on inference." Spatial Statistics 52 (2022): 100714.

See Also

simPopCustom, makePopIntegrationTab, adjustPopMat, simSPDE.

Examples

```
## Not run:
## In this script we will create 5km resolution pixellated grid over Kenya,
## and generate tables of estimated (both target and general) population
## totals at the area (e.g. Admin-1) and subarea (e.g. Admin-2) levels. Then
## we will use that to simulate populations of

# download Kenya GADM shapefiles from SUMMERdata github repository
githubURL <- paste0("https://github.com/paigejo/SUMMERdata/blob/main/data/",
                    "kenyaMaps.rda?raw=true")
tempDirectory = "~/"
mapsFilename = paste0(tempDirectory, "/kenyaMaps.rda")
if(!file.exists(mapsFilename)) {
  download.file(githubURL,mapsFilename)
}

# load it in
out = load(mapsFilename)
out
kenyaMesh <- fmesher::fm_as_fm(kenyaMesh)
adm1@data$NAME_1 = as.character(adm1@data$NAME_1)
adm1@data$NAME_1[adm1@data$NAME_1 == "Trans Nzoia"] = "Trans-Nzoia"
adm1@data$NAME_1[adm1@data$NAME_1 == "Elgeyo-Marakwet"] = "Elgeyo Marakwet"
adm2@data$NAME_1 = as.character(adm2@data$NAME_1)
adm2@data$NAME_1[adm2@data$NAME_1 == "Trans Nzoia"] = "Trans-Nzoia"
adm2@data$NAME_1[adm2@data$NAME_1 == "Elgeyo-Marakwet"] = "Elgeyo Marakwet"

# some Admin-2 areas have the same name
adm2@data$NAME_2 = as.character(adm2@data$NAME_2)
adm2@data$NAME_2[(adm2@data$NAME_1 == "Bungoma") &
                   (adm2@data$NAME_2 == "Lugari")] = "Lugari, Bungoma"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Kakamega") &
                   (adm2@data$NAME_2 == "Lugari")] = "Lugari, Kakamega"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Meru") &
                   (adm2@data$NAME_2 == "Igembe South")] = "Igembe South, Meru"
adm2@data$NAME_2[(adm2@data$NAME_1 == "Tharaka-Nithi") &
                   (adm2@data$NAME_2 == "Igembe South")] = "Igembe South, Tharaka-Nithi"

# The spatial area of unknown 8 is so small, it causes problems unless its removed or
```

```
# unioned with another subarea. Union it with neighboring Kakeguria:
newadm2 = adm2
unknown8I = which(newadm2$NAME_2 == "unknown 8")
newadm2$NAME_2[newadm2$NAME_2 %in% c("unknown 8", "Kapenguria")] <-
  "Kapenguria + unknown 8"
admin2.IDs <- newadm2$NAME_2

newadm2@data = cbind(newadm2@data, NAME_2OLD = newadm2@data$NAME_2)
newadm2@data$NAME_2OLD = newadm2@data$NAME_2
newadm2@data$NAME_2 = admin2.IDs
newadm2$NAME_2 = admin2.IDs
temp <- terra::aggregate(as(newadm2, "SpatVector"), by="NAME_2")

library(sf)
temp <- sf::st_as_sf(temp)
temp <- sf::as_Spatial(temp)

tempData = newadm2@data[-unknown8I,]
tempData = tempData[order(tempData$NAME_2),]
newadm2 <- sp::SpatialPolygonsDataFrame(temp, tempData, match.ID = F)
adm2 = newadm2

# download 2014 Kenya population density TIF file

githubURL <- paste0("https://github.com/paigejo/SUMMERdata/blob/main/data/",
                    "Kenya2014Pop/worldpop_total_1y_2014_00_00.tif?raw=true")
popTIFFilename = paste0(tempDirectory, "/worldpop_total_1y_2014_00_00.tif")
if(!file.exists(popTIFFilename)) {
  download.file(githubURL,popTIFFilename)
}

# load it in
pop = terra::rast(popTIFFilename)

east.lim = c(-110.6405, 832.4544)
north.lim = c(-555.1739, 608.7130)

## Construct poppsubKenya, a table of urban/rural general population totals
## in each subarea. Technically, this is not necessary since we can load in
## poppsubKenya via data(kenyaPopulationData). First, we will need to calculate
## the areas in km^2 of the areas and subareas

# use Lambert equal area projection of areas (Admin-1) and subareas (Admin-2)
midLon = mean(adm1@bbox[1,])
midLat = mean(adm1@bbox[2,])
p4s = paste0("+proj=laea +x_0=0 +y_0=0 +lon_0=", midLon,
             " +lat_0=", midLat, " +units=km")

adm1_sf = st_as_sf(adm1)
adm1proj_sf = st_transform(adm1_sf, p4s)
adm1proj = as(adm1proj_sf, "Spatial")

adm2_sf = st_as_sf(adm2)
```

```
adm2proj_sf = st_transform(adm2_sf, p4s)
adm2proj = as(adm2proj_sf, "Spatial")

# now calculate spatial area in km^2
admin1Areas = as.numeric(st_area(adm1proj_sf))
admin2Areas = as.numeric(st_area(adm2proj_sf))

areapaKenya = data.frame(area=adm1proj@data$NAME_1, spatialArea=admin1Areas)
areapsubKenya = data.frame(area=adm2proj@data$NAME_1, subarea=adm2proj@data$NAME_2,
                           spatialArea=admin2Areas)

# Calculate general population totals at the subarea (Admin-2) x urban/rural
# level and using 1km resolution population grid. Assign urbanicity by
# thresholding population density based on estimated proportion population
# urban/rural, making sure total area (Admin-1) urban/rural populations in
# each area matches poppaKenya.

# NOTE: the following function will typically take ~15-20 minutes. Can speed up
#       by setting km.res to be higher, but we recommend fine resolution for
#       this step, since it only needs to be done once. Instead of running
#       the code in the following if(FALSE) section,
#       you can simply run data(kenyaPopulationData)
if(FALSE){
  system.time(poppsubKenya <- getPoppsub(
    km.res=1, pop=pop, domain.map.dat=adm0,
    east.lim=east.lim, north.lim=north.lim, map.projection=projKenya,
    poppa = poppaKenya, areapa=areapaKenya, areapsub=areapsubKenya,
    area.map.dat=adm1, subarea.map.dat=adm2,
    areaNameVar = "NAME_1", subareaNameVar="NAME_2"))
}
data(kenyaPopulationData)

# Now generate a general population integration table at 5km resolution,
# based on subarea (Admin-2) x urban/rural population totals. This takes
# ~1 minute
pop.matKenya <- makePopIntegrationTab(
  km.res=5, pop=pop, domain.map.dat=adm0,
  east.lim=east.lim, north.lim=north.lim, map.projection=projKenya,
  poppa = poppaKenya, poppsub=poppsubKenya,
  area.map.dat = adm1, subarea.map.dat = adm2,
  areaNameVar = "NAME_1", subareaNameVar="NAME_2")

## Adjust pop.mat to be target (neonatal) rather than general population
## density. First create the target population frame
## (these numbers are based on IPUMS microcensus data)
mothersPerHouseholdUrb = 0.3497151
childrenPerMotherUrb = 1.295917
mothersPerHouseholdRur = 0.4787696
childrenPerMotherRur = 1.455222
targetPopPerStratumUrban = easpaKenya$HHUrb * mothersPerHouseholdUrb *
  childrenPerMotherUrb
targetPopPerStratumRural = easpaKenya$HHRur * mothersPerHouseholdRur *
  childrenPerMotherRur
```

```
easpaKenyaNeonatal = easpaKenya
easpaKenyaNeonatal$popUrb = targetPopPerStratumUrban
easpaKenyaNeonatal$popRur = targetPopPerStratumRural
easpaKenyaNeonatal$popTotal = easpaKenyaNeonatal$popUrb +
  easpaKenyaNeonatal$popRur
easpaKenyaNeonatal$pctUrb = 100 * easpaKenyaNeonatal$popUrb /
  easpaKenyaNeonatal$popTotal
easpaKenyaNeonatal$pctTotal =
  100 * easpaKenyaNeonatal$popTotal / sum(easpaKenyaNeonatal$popTotal)

# Generate the target population density by scaling the current
# population density grid at the Admin1 x urban/rural level
pop.matKenyaNeonatal = adjustPopMat(pop.matKenya, easpaKenyaNeonatal)

# Generate neonatal population table from the neonatal population integration
# matrix. This is technically not necessary for population simulation purposes,
# but is here for illustrative purposes
poppsubKenyaNeonatal = poppRegionFromPopMat(pop.matKenyaNeonatal,
                                            pop.matKenyaNeonatal$subarea)
poppsubKenyaNeonatal =
  cbind(subarea=poppsubKenyaNeonatal$region,
        area=adm2@data$NAME_1[match(poppsubKenyaNeonatal$region, adm2@data$NAME_2)],
        poppsubKenyaNeonatal[,-1])
print(head(poppsubKenyaNeonatal))

## Now we're ready to simulate neonatal populations along with neonatal
## mortality risks and prevalences

# use the following model to simulate the neonatal population based roughly
# on Paige et al. (2020) neonatal mortality modeling for Kenya.
beta0=-2.9 # intercept
gamma=-1 # urban effect
rho=(1/3)^2 # spatial variance
eff.range = 400 # effective spatial range in km
sigma.epsilon=sqrt(1/2.5) # cluster (nugget) effect standard deviation

# Run a simulation! This produces multiple dense nEA x nsim and nPixel x nsim
# matrices. In the future sparse matrices and chunk by chunk computations
# may be incorporated.
simPop = simPopSPDE(nsim=1, easpa=easpaKenyaNeonatal,
                    pop.mat=pop.matKenya, target.pop.mat=pop.matKenyaNeonatal,
                    poppsub=poppsubKenya, spde.mesh=kenyaMesh,
                    marg.var=rho, sigma.epsilon=sigma.epsilon,
                    gamma=gamma, eff.range=eff.range, beta0=beta0,
                    seed=12, inla.seed=12, n.HH.sampled=25,
                    stratify.by.urban=TRUE, subarea.level=TRUE,
                    do.fine.scale.risk=TRUE, do.smooth.risk=TRUE,
                    min1.per.subarea=TRUE)

# get average absolute percent error relative to fine scale prevalence at Admin-2 level
tempDat = simPop$subareaPop$aggregationResults[c("region", "pFineScalePrevalence",
                                                 "pFineScaleRisk", "pSmoothRisk")]
100*mean(abs(tempDat$pFineScalePrevalence - tempDat$pFineScaleRisk) /
```

```
          tempDat$pFineScalePrevalence)
100*mean(abs(tempDat$pFineScalePrevalence - tempDat$pSmoothRisk) /
          tempDat$pFineScalePrevalence)
100*mean(abs(tempDat$pFineScaleRisk - tempDat$pSmoothRisk) /
          tempDat$pFineScalePrevalence)

# verify number of EAs per area and subarea
cbind(aggregate(simPop$eaPop$ea.samples[,1], by=list(area=pop.matKenya$area), FUN=sum),
      trueNumEAs=easpaKenya$EATotal[order(easpaKenya$area)])
aggregate(simPop$eaPop$ea.samples[,1], by=list(area=pop.matKenya$subarea), FUN=sum)

## plot simulated population
# directory for plotting
# (mapPlot takes longer when it doesn't save to a file)
tempDirectory = "~/"

# pixel level plots. Some pixels have no simulated EAs, in which case they will be
# plotted as white. Expected noisy looking plots of fine scale risk and prevalence
# due to EAs being discrete, as compared to a very smooth plot of smooth risk.
zlim = c(0, quantile(probs=.995, c(simPop$pixelPop$pFineScalePrevalence,
                                   simPop$pixelPop$pFineScaleRisk,
                                   simPop$pixelPop$pSmoothRisk), na.rm=TRUE))
par(mfrow=c(2,2))
plot(adm2, asp=1)
points(simPop$eaPop$eaDatList[[1]]$lon, simPop$eaPop$eaDatList[[1]]$lat, pch=".", col="blue")
plot(adm2, asp=1)
fields::quilt.plot(pop.matKenya$lon, pop.matKenya$lat, simPop$pixelPop$pFineScalePrevalence,
          zlim=zlim, add=TRUE, FUN=function(x) {mean(x, na.rm=TRUE)})
plot(adm2, asp=1)
fields::quilt.plot(pop.matKenya$lon, pop.matKenya$lat, simPop$pixelPop$pFineScaleRisk,
          zlim=zlim, add=TRUE, FUN=function(x) {mean(x, na.rm=TRUE)})
fields::quilt.plot(pop.matKenya$lon, pop.matKenya$lat, simPop$pixelPop$pSmoothRisk,
          zlim=zlim, FUN=function(x) {mean(x, na.rm=TRUE)}, asp=1)
plot(adm2, add=TRUE)

range(simPop$eaPop$eaDatList[[1]]$N)

# areal (Admin-1) level: these results should look essentially identical

tempDat = simPop$areaPop$aggregationResults[c("region", "pFineScalePrevalence",
                                              "pFineScaleRisk", "pSmoothRisk")]
mapPlot(tempDat,
        variables=c("pFineScalePrevalence", "pFineScaleRisk", "pSmoothRisk"),
        geo=adm1, by.geo="NAME_1", by.data="region", is.long=FALSE)

# subareal (Admin-2) level: these results should look subtley different
# depending on the type of prevalence/risk considered
tempDat = simPop$subareaPop$aggregationResults[c("region", "pFineScalePrevalence",
                                                 "pFineScaleRisk", "pSmoothRisk")]
mapPlot(tempDat,
        variables=c("pFineScalePrevalence", "pFineScaleRisk", "pSmoothRisk"),
        geo=adm2, by.geo="NAME_2", by.data="region", is.long=FALSE)
```

```
## End(Not run)
```

---

simPopInternal *Internal functions for population simulation*

Description

Functions for calculating valuable quantities and for drawing from important distributions for population simulation.

Usage

```
getExpectedNperEA(
  easpa,
  pop.mat,
  level = c("grid", "EA"),
  pixel.index.mat = NULL
)

getSortIndices(
  i,
  urban = TRUE,
  pop.mat,
  stratify.by.urban = TRUE,
  validation.pixel.I = NULL
)

rStratifiedMultnomial(n, pop.mat, easpa, stratify.by.urban = TRUE)

rStratifiedMultnomialBySubarea(
  n,
  pop.mat,
  easpa,
  stratify.by.urban = TRUE,
  poppsub = NULL,
  min1.per.subarea = TRUE,
  min.sample = 1
)

rMyMultinomial(
  n,
  i,
  stratify.by.urban = TRUE,
  urban = TRUE,
  pop.mat = NULL,
  easpa = NULL,
  min1.per.subarea = FALSE,
```

```
  method = c("mult1", "mult", "indepMH"),
  min.sample = 1
)

rMyMultinomialSubarea(
  n,
  i,
  easpsub,
  stratify.by.urban = TRUE,
  urban = TRUE,
  pop.mat = NULL
)

rmultinom1(
  n = 1,
  size,
  prob,
  max.size = 8000 * 8000,
  method = c("mult1", "mult", "indepMH"),
  verbose = FALSE,
  min.sample = 100,
  max.expected.size.before.switch = 1000 * 1e+07,
  init = NULL,
  burnin = floor(n/4),
  filter.every = 10,
  zero.prob.zero.samples = TRUE,
  allow.size.less.than.K = FALSE
)

sampleNMultilevelMultinomial(
  ndraws = ncol(pixel.index.mat),
  pixel.index.mat = NULL,
  urban.mat = NULL,
  area.mat = NULL,
  easpa.list,
  pop.mat,
  stratify.by.urban = TRUE,
  verbose = TRUE,
  return.EA.info = FALSE,
  min.HH.per.EA = 25,
  fix.HH.per.EA = NULL,
  fix.pop.per.HH = NULL
)

sampleNMultilevelMultinomialFixed(
  clusters.per.pixel,
  ndraws = ncol(pixel.indices),
  pixel.indices = NULL,
```

```
  urbanVals = NULL,
  areaVals = NULL,
  easpa,
  pop.mat,
  stratify.by.urban = TRUE,
  verbose = TRUE
)
```

Arguments

easpa	Census frame. See simPopCustom for details
pop.mat	data.frame of pixellated grid of population densities. See simPopCustom for details
level	Whether to calculate results at the integration grid or EA level
pixel.index.mat	Matrix of pixel indices associated with each EA and draw. Not required by getExpectedNperEA unless level == "EA"
i	Index
urban	If TRUE, calculate only for urban part of the area. If FALSE, for only rural part
stratify.by.urban	whether or not to stratify calculations by urban/rural classification
validation.pixel.I	CURRENTLY FOR TESTING PURPOSES ONLY a set of indices of pixels for which we want to simulate populations (used for pixel level validation)
n	Number of samples
poppsub	Population per subarea. See simPopCustom for details
min1.per.subarea	Whether or not to ensure there is at least 1 EA per subarea. See simPopCustom for details
min.sample	The minimum number of samples per chunk of samples for truncated multinomial sampling. Defaults to 1
method	If min1.per.subarea is TRUE, the sampling method for the truncated multinomial to use with rmulitnom1. rmultinom1 automatically switches between them depending on the number of expected samples. The methods are: **mult1** rejection sampling from multinomial plus 1 in each category **mult** rejection sampling from multinomial if any category has zero count **indepMH** independent Metropolis-Hastings using multinomial plus 1 distribution as proposal
easpsub	This could either be total EAs per subarea, or subarea crossed with urban or rural if stratify.by.urban is TRUE
size	Multinomial size parameter. See rmultinom
prob	Multinomial probability vector parameter. See rmultinom
max.size	The maximum number of elements in a matrix drawn from the proposal distribution per sample chunk.

verbose	Whether to print progress as the function proceeds
max.expected.size.before.switch	Max expected number of samples / k, the number of categories, before switching method
init	Initial sample if method is indepMH
burnin	Number of initial samples before samples are collected if method is indepMH
filter.every	Store only every filter.every samples if method is iindepMH
zero.prob.zero.samples	If TRUE, set samples for parts of prob vector that are zero to zero. Otherwise they are set to one.
allow.size.less.than.K	If TRUE, then if size < the number of categories (k), returns matrix where each column is vector of size ones and k - size zeros. If FALSE, throws an error if size < k
ndraws	Number of draws
urban.mat	Matrix of urbanicities associated with each EA and draw
area.mat	Matrix of areas associated with each EA and draw
easpa.list	A list of length n with each element being of the format of easpa giving the number of households and EAs per stratum. It is assumed that the number of EAs per stratum is the same in each list element. If easpa.list is a data frame, number of households per stratum is assumed constant
return.EA.info	Whether a data frame at the EA level is desired
min.HH.per.EA	The minimum number of households per EA (defaults to 25, since that is the number of households sampled per DHS cluster)
fix.HH.per.EA	If not NULL, the fixed number of households per EA
fix.pop.per.HH	If not NULL, the fixed target population per household
clusters.per.pixel	CURRENTLY FOR TESTING PURPOSES ONLY a vector of length nIntegrationPoints specifying the number of clusters per pixel if they are fixed
pixel.indices	A nEA x n matrix of pixel indices associated with each EA per simulation/draw
urbanVals	A nEA x n matrix of urbanicities associated with each EA per simulation/draw
areaVals	A nEA x n matrix of area names associated with each EA per simulation/draw

Details

[Experimental]

Functions

- getExpectedNperEA(): Calculates expected denominator per enumeration area.
- getSortIndices(): For recombining separate multinomials into the draws over all grid points

- `rStratifiedMultnomial()`: Gives nIntegrationPoints x n matrix of draws from the stratified multinomial with values corresponding to the value of |C^g| for each pixel, g (the number of EAs/pixel)
- `rStratifiedMultnomialBySubarea()`: Gives nIntegrationPoints x n matrix of draws from the stratified multinomial with values
- `rMyMultinomial()`:
- `rMyMultinomialSubarea()`:
- `rmultinom1()`: Random (truncated) multinomial draws conditional on the number of each type being at least one
- `sampleNMultilevelMultinomial()`: Take multilevel multinomial draws first from joint distribution of number of households per EA given the total per stratum, and then from the joint distribution of the total target population per household given the total per stratum
- `sampleNMultilevelMultinomialFixed()`: Same as sampleNMultilevelMultinomial, except the number of EAs per pixel is fixed

---

`simSPDE` *Simulate from the SPDE spatial model*

Description

Generates nCoords x nsim matrix of simulated values of the SPDE spatial process

Usage

```
simSPDE(
  coords,
  nsim = 1,
  mesh,
  eff.range = (max(coords[, 1]) - min(coords[, 1]))/3,
  marg.var = 1,
  inla.seed = 0L
)
```

Arguments

`coords`	2 column matrix of spatial coordinates at which to simulate the spatial process
`nsim`	number of draws from the SPDE model
`mesh`	SPDE mesh
`eff.range`	effective spatial range
`marg.var`	marginal variance of the spatial process
`inla.seed`	seed input to inla.qsample. 0L sets seed intelligently, positive value sets a specific seed, negative value keeps existing RNG

Details

[Experimental]

Author(s)

John Paige

References

Lindgren, F., Rue, H., Lindström, J., 2011. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic differential equation approach (with discussion). Journal of the Royal Statistical Society, Series B 73, 423–498.

Examples

```
## Not run:
set.seed(123)
require(INLA)
coords = matrix(runif(10*2), ncol=2)
mesh = inla.mesh.2d(loc.domain=cbind(c(0, 0, 1, 1), c(0, 1, 0, 1)),
  n=3000, max.n=5000, max.edge=c(.01, .05), offset=-.1)
simVals = simSPDE(coords, nsim=1, mesh, eff.range=.2, inla.seed=1L)

## End(Not run)
```

---

smoothArea *Small area estimation via basic area level model*

Description

Generates small area estimates by smoothing direct estimates using an area level model

Usage

```
smoothArea(
  formula,
  domain,
  design = NULL,
  adj.mat = NULL,
  X.domain = NULL,
  direct.est = NULL,
  domain.size = NULL,
  transform = c("identity", "logit", "log"),
  pc.u = 1,
  pc.alpha = 0.01,
  pc.u.phi = 0.5,
  pc.alpha.phi = 2/3,
```

```
  level = 0.95,
  n.sample = 250,
  var.tol = 1e-10,
  return.samples = F
)
```

Arguments

formula	An object of class 'formula' describing the model to be fitted. If direct.est is specified, the right hand side of the formula is not necessary.
domain	One-sided formula specifying factors containing domain labels
design	An object of class "svydesign" containing the data for the model
adj.mat	Adjacency matrix with rownames matching the domain labels. If set to NULL, the IID spatial effect will be used.
X.domain	Data frame of areal covariates. One of the column names needs to match the name of the domain variable, in order to be linked to the data input. Currently only supporting time-invariant covariates.
direct.est	Data frame of direct estimates, with first column containing the domain variable, second column containing direct estimate, and third column containing the variance of direct estimate.
domain.size	Data frame of domain sizes. One of the column names needs to match the name of the domain variable, in order to be linked to the data input and there must be a column names 'size' containing domain sizes.
transform	Optional transformation applied to the direct estimates before fitting area level model. The default option is no transformation, but logit and log are implemented.
pc.u	Hyperparameter U for the PC prior on precisions. See the INLA documentation for more details on the parameterization.
pc.alpha	Hyperparameter alpha for the PC prior on precisions.
pc.u.phi	Hyperparameter U for the PC prior on the mixture probability phi in BYM2 model.
pc.alpha.phi	Hyperparameter alpha for the PC prior on the mixture probability phi in BYM2 model.
level	The specified level for the posterior credible intervals
n.sample	Number of draws from posterior used to compute summaries
var.tol	Tolerance parameter; if variance of an area's direct estimator is below this value, that direct estimator is dropped from model
return.samples	If TRUE, return matrix of posterior samples of area level quantities

Value

A svysae object

Examples

```
## Not run:
data(DemoData2)
data(DemoMap2)
library(survey)
des0 <- svydesign(ids = ~clustid+id, strata = ~strata,
                  weights = ~weights, data = DemoData2, nest = TRUE)
Xmat <- aggregate(age~region, data = DemoData2, FUN = mean)

# EXAMPLE 1: Continuous response model
cts.res <- smoothArea(tobacco.use ~ 1,
                      domain = ~region,
                      design = des0,
                      adj.mat = DemoMap2$Amat,
                      pc.u = 1,
                      pc.alpha = 0.01,
                      pc.u.phi = 0.5,
                      pc.alpha.phi = 2/3)

# EXAMPLE 2: Including area level covariates
cts.cov.res <- smoothArea(tobacco.use ~ age,
                          domain = ~region,
                          design = des0,
                          adj.mat = DemoMap2$Amat,
                          X.domain = Xmat,
                          pc.u = 1,
                          pc.alpha = 0.01,
                          pc.u.phi = 0.5,
                          pc.alpha.phi = 2/3)

# EXAMPLE 3: Binary response model
bin.res <- smoothArea(tobacco.use ~ 1,
                      domain = ~region,
                      design = des0,
                      adj.mat = DemoMap2$Amat,
                      transform = "logit",
                      pc.u = 1,
                      pc.alpha = 0.01,
                      pc.u.phi = 0.5,
                      pc.alpha.phi = 2/3)

# EXAMPLE 4: Including area level covariates in binary response model
bin.cov.res <- smoothArea(tobacco.use ~ age,
                          domain = ~region,
                          design = des0,
                          adj.mat = DemoMap2$Amat,
                          transform = "logit",
                          X.domain = Xmat,
                          pc.u = 1,
                          pc.alpha = 0.01,
                          pc.u.phi = 0.5,
                          pc.alpha.phi = 2/3)
```

```
## End(Not run)
```

`smoothCluster` *Cluster-level space-time smoothing models for mortality rates*

Description

The function `smoothCluster` replace the previous function name `fitINLA2` (before version 1.0.0).

Usage

```
smoothCluster(
  data,
  X = NULL,
  family = c("betabinomial", "binomial")[1],
  age.group = c("0", "1-11", "12-23", "24-35", "36-47", "48-59"),
  age.groups = deprecated(),
  age.n = c(1, 11, 12, 12, 12, 12),
  age.time.group = c(1, 2, 3, 3, 3, 3),
  age.strata.fixed.group = c(1, 2, 3, 4, 5, 6),
  time.model = c("rw1", "rw2", "ar1")[2],
  st.time.model = NULL,
  Amat,
  bias.adj = NULL,
  bias.adj.by = NULL,
  formula = NULL,
  year.label,
  year_label = deprecated(),
  type.st = 4,
  survey.effect = FALSE,
  linear.trend = TRUE,
  common.trend = FALSE,
  strata.time.effect = FALSE,
  hyper = "pc",
  pc.u = 1,
  pc.alpha = 0.01,
  pc.u.phi = 0.5,
  pc.alpha.phi = 2/3,
  pc.u.cor = 0.7,
  pc.alpha.cor = 0.9,
  pc.st.u = NA,
  pc.st.alpha = NA,
  pc.st.slope.u = NA,
  pc.st.slope.alpha = NA,
  overdisp.mean = 0,
  overdisp.prec = 0.4,
```

```
  options = list(config = TRUE),
  control.inla = list(strategy = "adaptive", int.strategy = "auto"),
  control.fixed = list(),
  verbose = FALSE,
  geo = NULL,
  rw = NULL,
  ar = NULL,
  st.rw = NULL,
  age.rw.group = NULL,
  ...
)
```

Arguments

data	count data of person-months with the following columns • cluster: cluster ID • years: time period • region: region of the cluster • strata: stratum of the cluster • age: age group corresponding to the row • total: total number of person-month in this age group, stratum, cluster, and period • Y: total number of deaths in this age group, stratum, cluster, and period
X	Covariate matrix. It must contain either a column with name "region", or a column with name "years", or both. The covariates must not have missing values for all regions (if varying in space) and all time periods (if varying in time). The rest of the columns are treated as covariates in the mean model.
family	family of the model. This can be either binomial (with logistic normal prior), betabiniomial.
age.group	a character vector of age groups in increasing order.
age.groups	**[Deprecated]** replaced by age.group
age.n	number of months in each age groups in the same order.
age.time.group	vector indicating grouping of the ages groups in the temporal model. For example, if each age group is assigned a different temporal component, then set age.rw.group to c(1:length(age.group)); if all age groups share the same random walk component, then set age.rw.group to a rep(1, length(age.group)). The default for 6 age groups is c(1,2,3,3,3,3), which assigns a separate temporal trend to the first two groups and a common random walk for the rest of the age groups. The vector should contain values starting from 1. This argument replaces the previous age.rw.group argument.
age.strata.fixed.group	vector indicating grouping of the ages groups for different strata in the intercept. The default is c(1:length(age.group)), which correspond to each age group within each stratum receives a separate intercept. If several age groups are specified to be the same value in this vector, the stratum specific deviation from

the baseline is assumed to be the same for these age groups. For example, if `age.strata.fixed.group = c(1, 2, 3, 3, 3, 3)`, then the intercept part of the linear predictor consists of 6 overall age-specific intercepts and 3 set of strata effects (where a base stratum is chosen internally), for age groups 1, 2, and the rest respectively. Notice that this argument does not control the linear temporal trends (which is also parameterized as fixed effect, but determined by the `age.rw.group` argument). The vector should contain values starting from 1.

More specific examples: (1) if each age group is assigned a different intercept, then set age.strata.fixed.group to c(1:length(age.group)) (2) if all age groups share the same intercept, then set age.strata.fixed.group to a rep(1, length(age.group)). The default for 6 age groups is the former. (3) If each temporal trend is associated with its own intercept, set it to be the same as `age.rw.group`.

`time.model` Model for the main temporal trend, can be rw1, rw2, ar1, or NULL (for spatial-only smoothing). Default to be rw2. For ar1 main effect, a linear slope is also added with time scaled to be between -0.5 to 0.5, i.e., the slope coefficient represents the total change between the first year and the last year in the projection period on the logit scale.

`st.time.model` Temporal component model for the interaction term, can be rw1, rw2, or ar1. Default to be the same as time.model unless specified otherwise. The default does not include region-specific random slopes. They can be added to the interaction term by specifying `pc.st.slope.u` and `pc.st.slope.alpha`.

`Amat` Adjacency matrix for the regions

`bias.adj` the ratio of unadjusted mortality rates or age-group-specific hazards to the true rates or hazards. It needs to be a data frame that can be merged to thee outcome, i.e., with the same column names for time periods (for national adjustment), or time periods and region (for subnational adjustment). The column specifying the adjustment ratio should be named "ratio".

`bias.adj.by` vector of the column names specifying how to merge the bias adjustment to the count data. For example, if bias adjustment factor is provided in bias.adj for each region and time, then bias.adj.by should be `c("region", "time")`.

`formula` INLA formula. See vignette for example of using customized formula.

`year.label` string vector of year names

`year_label` **[Deprecated]** replaced by year.label

`type.st` type for space-time interaction

`survey.effect` logical indicator whether to include a survey fixed effect. If this is set to TRUE, there needs to be a column named 'survey' in the input data frame. In prediction, this effect term will be set to 0.

`linear.trend` logical indicator whether a linear trend is added to the temporal main effect. If the temporal main effect is RW2, then it will be forced to FALSE. Default is TRUE.

`common.trend` logical indicator whether all age groups and strata share the same linear trend in the temporal main effect.

`strata.time.effect`
logical indicator whether to include strata specific temporal trends.

hyper	Deprecated. which hyperpriors to use. Only supports PC prior ("pc").
pc.u	hyperparameter U for the PC prior on precisions.
pc.alpha	hyperparameter alpha for the PC prior on precisions.
pc.u.phi	hyperparameter U for the PC prior on the mixture probability phi in BYM2 model.
pc.alpha.phi	hyperparameter alpha for the PC prior on the mixture probability phi in BYM2 model.
pc.u.cor	hyperparameter U for the PC prior on the autocorrelation parameter in the AR prior, i.e. Prob(cor > pc.u.cor) = pc.alpha.cor.
pc.alpha.cor	hyperparameter alpha for the PC prior on the autocorrelation parameter in the AR prior.
pc.st.u	hyperparameter U for the PC prior on precisions for the interaction term.
pc.st.alpha	hyperparameter alpha for the PC prior on precisions for the interaction term.
pc.st.slope.u	hyperparameter U for the PC prior on precisions for the area-level random slope. If both pc.st.slope.u and pc.st.slope.alpha are not NA, an area-level random slope with iid prior will be added to the model. The parameterization of the random slope is so that Prob(|beta| > pc.st.slope.u) = pc.st.slope.alpha, where time covariate is rescaled to be -0.5 to 0.5, so that the random slope can be interpreted as the total deviation from the main trend from the first year to the last year to be projected, on the logit scale.
pc.st.slope.alpha	hyperparameter alpha for the PC prior on precisions for the area-level random slope. See above for the parameterization.
overdisp.mean	hyperparameter for the betabinomial likelihood. Mean of the over-dispersion parameter on the logit scale.
overdisp.prec	hyperparameter for the betabinomial likelihood. Precision of the over-dispersion parameter on the logit scale.
options	list of options to be passed to control.compute() in the inla() function.
control.inla	list of options to be passed to control.inla() in the inla() function. Default to the "adaptive" integration strategy.
control.fixed	list of options to be passed to control.fixed() in the inla() function.
verbose	logical indicator to print out detailed inla() intermediate steps.
geo	Deprecated. Spatial polygon file, legacy parameter from previous versions of the package.
rw	Deprecated. Take values 0, 1 or 2, indicating the order of random walk. If rw = 0, the autoregressive process is used instead of the random walk in the main trend. See the description of the argument ar for details.
ar	Deprecated. Order of the autoregressive component. If ar is specified to be positive integer, the random walk components will be replaced by AR(p) terms in the interaction part. The main temporal trend remains to be random walk of order rw unless rw = 0.
st.rw	Deprecated. Take values 1 or 2, indicating the order of random walk for the interaction term. If not specified, it will take the same order as the argument rw in the main effect. Notice that this argument is only used if ar is set to 0.

age.rw.group	Deprecated. Legacy parameter replaced by age.time.group.
...	arguments to be passed to the inla() function call.

Value

INLA model fit using the provided formula, country summary data, and geographic data

Author(s)

Zehang Richard Li

See Also

getDirect

Examples

```
## Not run:
library(dplyr)
data(DemoData)
# Create dataset of counts
counts.all <- NULL
for(i in 1:length(DemoData)){
  counts <- getCounts(DemoData[[i]][, c("clustid", "time", "age", "died",
                                        "region", "strata")],
            variables = 'died', by = c("age", "clustid", "region",
                                       "time", "strata"))
  counts <- counts %>% mutate(cluster = clustid, years = time, Y=died)
  counts$strata <- gsub(".*\\.","",counts$strata)
  counts$survey <- names(DemoData)[i]
  counts.all <- rbind(counts.all, counts)
}

# fit cluster-level model on the periods
periods <- levels(DemoData[[1]]$time)
fit <- smoothCluster(data = counts.all,
     Amat = DemoMap$Amat,
     time.model = "rw2",
     st.time.model = "rw1",
     strata.time.effect =  TRUE,
     survey.effect = TRUE,
     family = "betabinomial",
     year.label = c(periods, "15-19"))
summary(fit)
est <- getSmoothed(fit, nsim = 1000)
plot(est$stratified, plot.CI=TRUE) + ggplot2::facet_wrap(~strata)

# fit cluster-level space-time model with covariate
# notice without projected covariates, we use periods up to 10-14 only
# construct a random covariate matrix for illustration
periods <- levels(DemoData[[1]]$time)
X <- expand.grid(years = periods,
```

```
       region = unique(counts.all$region))
X$X1 <- rnorm(dim(X)[1])
X$X2 <- rnorm(dim(X)[1])
fit.covariate <- smoothCluster(data = counts.all,
   X = X,
     Amat = DemoMap$Amat,
     time.model = "rw2",
     st.time.model = "rw1",
     strata.time.effect =  TRUE,
     survey.effect = TRUE,
     family = "betabinomial",
     year.label = c(periods))
est <- getSmoothed(fit.covariate, nsim = 1000)

# fit cluster-level model for one time point only
# i.e., space-only model
fit.sp <- smoothCluster(data = subset(counts.all, time == "10-14"),
     Amat = DemoMap$Amat,
     time.model = NULL,
     survey.effect = TRUE,
     family = "betabinomial")
summary(fit.sp)
est <- getSmoothed(fit.sp, nsim = 1000)
plot(est$stratified, plot.CI = TRUE) + ggplot2::facet_wrap(~strata)

# fit cluster-level model for one time point and covariate
# construct a random covariate matrix for illustration
X <- data.frame(region = unique(counts.all$region),
      X1 = c(1, 2, 2, 1),
      X2 = c(1, 1, 1, 2))
fit.sp.covariate <- smoothCluster(data = subset(counts.all, time == "10-14"),
     X = X,
     Amat = DemoMap$Amat,
     time.model = NULL,
     survey.effect = TRUE,
     family = "betabinomial")
summary(fit.sp.covariate)
est <- getSmoothed(fit.sp.covariate, nsim = 1000)

## End(Not run)
```

`smoothDirect` *Smoothed direct estimates for mortality rates*

Description

The function `smoothDirect` replaces the previous function name `fitINLA` (before version 1.0.0).

Usage

```
smoothDirect(
  data,
  Amat,
  formula = NULL,
  time.model = c("rw1", "rw2", "ar1")[2],
  st.time.model = NULL,
  year.label,
  year_label = deprecated(),
  year.range = c(1980, 2014),
  year_range = deprecated(),
  is.yearly = TRUE,
  m = 5,
  type.st = 1,
  survey.effect = FALSE,
  hyper = c("pc", "gamma")[1],
  pc.u = 1,
  pc.alpha = 0.01,
  pc.u.phi = 0.5,
  pc.alpha.phi = 2/3,
  pc.u.cor = 0.7,
  pc.alpha.cor = 0.9,
  pc.st.u = NA,
  pc.st.alpha = NA,
  control.compute = list(dic = TRUE, mlik = TRUE, cpo = TRUE, openmp.strategy =
    "default", config = TRUE),
  control.inla = list(strategy = "adaptive", int.strategy = "auto"),
  control.fixed = list(),
  verbose = FALSE,
  geo = NULL,
  rw = NULL,
  ar = NULL,
  options = NULL
)
```

Arguments

data	Combined dataset
Amat	Adjacency matrix for the regions
formula	INLA formula. See vignette for example of using customized formula.
time.model	Model for the main temporal trend, can be rw1, rw2, or ar1. ar1 is not implemented for yearly model with period data input. Default to be rw2. For ar1 main effect, a linear slope is also added with time scaled to be between -0.5 to 0.5, i.e., the slope coefficient represents the total change between the first year and the last year in the projection period on the logit scale.
st.time.model	Temporal component model for the interaction term, can be rw1, rw2, or ar1. ar1 is not implemented for yearly model with period data input. Default to be

	the same as time.model unless specified otherwise. For ar1 interaction model, region-specific random slopes are currently not implemented.
year.label	string vector of year names
year_label	**[Deprecated]** replaced by year.label
year.range	Entire range of the years (inclusive) defined in year.label.
year_range	**[Deprecated]** replaced by year.range
is.yearly	Logical indicator for fitting yearly or period model.
m	Number of years in each period.
type.st	type for space-time interaction
survey.effect	logical indicator whether to include a survey iid random effect. If this is set to TRUE, there needs to be a column named 'survey' in the input data frame. In prediction, this random effect term will be set to 0. Notice this survey effect is implemented according to the Merter et al. (2015) model, and differently compared to the smoothCluster() function.
hyper	which hyperpriors to use. Default to be using the PC prior ("pc").
pc.u	hyperparameter U for the PC prior on precisions.
pc.alpha	hyperparameter alpha for the PC prior on precisions.
pc.u.phi	hyperparameter U for the PC prior on the mixture probability phi in BYM2 model.
pc.alpha.phi	hyperparameter alpha for the PC prior on the mixture probability phi in BYM2 model.
pc.u.cor	hyperparameter U for the PC prior on the autocorrelation parameter in the AR prior, i.e. Prob(cor > pc.u.cor) = pc.alpha.cor.
pc.alpha.cor	hyperparameter alpha for the PC prior on the autocorrelation parameter in the AR prior.
pc.st.u	hyperparameter U for the PC prior on precisions for the interaction term.
pc.st.alpha	hyperparameter alpha for the PC prior on precisions for the interaction term.
control.compute	list of options to be passed to control.compute() in the inla() function. The default argument saves the internal objects created by INLA for posterior sampling later. If the fitted object is too large in size and there is no need to perform joint posterior sampling from the model (only used in benchmarking), this argument can be set to `control.compute = list(config = FALSE)` to reduce the size of the fitted object.
control.inla	list of options to be passed to control.inla() in the inla() function. Default to the "adaptive" integration strategy.
control.fixed	list of options to be passed to control.fixed() in the inla() function.
verbose	logical indicator to print out detailed inla() intermediate steps.
geo	Deprecated.
rw	Deprecated.
ar	Deprecated.
options	Deprecated.

Value

List of fitted object

Author(s)

Zehang Richard Li

References

Li, Z., Hsiao, Y., Godwin, J., Martin, B. D., Wakefield, J., Clark, S. J., & with support from the United Nations Inter-agency Group for Child Mortality Estimation and its technical advisory group. (2019). *Changes in the spatial distribution of the under-five mortality rate: Small-area analysis of 122 DHS surveys in 262 subregions of 35 countries in Africa.* PloS one, 14(1), e0210645.

Mercer, L. D., Wakefield, J., Pantazis, A., Lutambi, A. M., Masanja, H., & Clark, S. (2015). *Space-time smoothing of complex survey data: small area estimation for child mortality.* The annals of applied statistics, 9(4), 1889.

See Also

getDirect

Examples

```
## Not run:
  years <- levels(DemoData[[1]]$time)
  # obtain direct estimates
  data_multi <- getDirectList(births = DemoData, years = years,
  regionVar = "region",  timeVar = "time", clusterVar = "~clustid+id",
  ageVar = "age", weightsVar = "weights", geo.recode = NULL)
  data <- aggregateSurvey(data_multi)

  #  national model
  years.all <- c(years, "15-19")
  fit1 <- smoothDirect(data = data, Amat = NULL,
  year.label = years.all, year.range = c(1985, 2019),
  time.model = 'rw2', m = 5, control.compute = list(config =TRUE))
  out1 <- getSmoothed(fit1)
  plot(out1)

  #  subnational model
  fit2 <- smoothDirect(data = data, Amat = DemoMap$Amat,
  year.label = years.all, year.range = c(1985, 2019),
  time.model = 'rw2', m = 5, type.st = 4)
  out2 <- getSmoothed(fit2)
  plot(out2)

  #  subnational space-only model for one period
  fit3 <- smoothDirect(data = subset(data, years == "10-14"),
           time.model = NULL, Amat = DemoMap$Amat)
  out3 <- getSmoothed(fit3)
  plot(out3, plot.CI = TRUE)
```

```
## End(Not run)
```

`smoothSurvey` *Fit space-time smoothing models for a binary outcome from complex surveys.*

Description

This function calculates the direct estimates by region and fit a simple spatial smoothing model to the direct estimates adjusting for survey design. Normal or binary variables are currently supported. For binary variables, the logit transformation is performed on the direct estimates of probabilities, and a Gaussian additive model is fitted on the logit scale using INLA.

Usage

```
smoothSurvey(
  data,
  geo = NULL,
  Amat = NULL,
  region.list = NULL,
  X = NULL,
  X.unit = NULL,
  responseType = deprecated(),
  response.type = c("binary", "gaussian")[1],
  responseVar,
  strataVar = "strata",
  weightVar = "weights",
  regionVar = "region",
  clusterVar = "~v001+v002",
  pc.u = 1,
  pc.alpha = 0.01,
  pc.u.phi = 0.5,
  pc.alpha.phi = 2/3,
  CI = 0.95,
  formula = NULL,
  timeVar = NULL,
  time.model = c("rw1", "rw2")[1],
  include_time_unstruct = deprecated(),
  include.time.unstruct = FALSE,
  type.st = 1,
  direct.est = NULL,
  direct.est.var = NULL,
  is.unit.level = FALSE,
  is.agg = FALSE,
  strataVar.within = NULL,
  totalVar = NULL,
```

```
  weight.strata = NULL,
  nsim = 1000,
  save.draws = FALSE,
  smooth = TRUE,
  ...
)
```

Arguments

data	The input data frame. The input data with column of the response variable (responseVar), region ID (regionVar), stratification within region (strataVar), and cluster ID (clusterVar). • For area-level model, the data frame consist of survey observations and corresponding survey weights (weightVar). • For unit-level model and is.agg = FALSE, the data frame should consist of aggregated counts by clusters (for binary responses), or any cluster-level response (for continuous response). For binary response (response.type = 'binary'), the beta-binomial model will be fitted for cluster-level counts. For continuous response (response.type = 'gaussian'), a Gaussian smoothing model will be fitted on the cluster-level response. • For unit-level model and is.agg = TRUE, the data frame should be the same as in the area-level model. For binary response (response.type = 'binary'), the beta-binomial model will be fitted for cluster-level counts aggregated internally. For continuous response (response.type = 'gaussian'), the nested error model will be fitted on unit-level response.
geo	Deprecated argument from early versions.
Amat	Adjacency matrix for the regions. If set to NULL, the IID spatial effect will be used.
region.list	a vector of region names. Only used when IID model is used and the adjacency matrix not specified. This allows the output to include regions with no sample in the data. When the spatial adjacency matrix is specified, the column names of the adjacency matrix will be used to determine region.list. If set to NULL, all regions in the data are used.
X	Areal covariates data frame. One of the column name needs to match the regionVar specified in the function call, in order to be linked to the data input. Currently only supporting time-invariant region-level covariates.
X.unit	Column names of unit-level covariates. When X.unit is specified, a nested error model will be fitted with unit-level IID noise, and area-level predictions are produced by plugging in the covariate specified in the X argument. When X is not specified, the empirical mean of each covariate will be used. This is only implemented for continuous response with the Gaussian likelihood model and unit-level model.
responseType	**[Deprecated]** The argument has been renamed into response.type.
response.type	Type of the response variable, currently supports 'binary' (default with logit link function) or 'gaussian'.
responseVar	the response variable

`strataVar` the strata variable used in the area-level model.

`weightVar` the weights variable

`regionVar` Variable name for region.

`clusterVar` Variable name for cluster. For area-level model, this should be a formula for cluster in survey design object, e.g., '~clusterID + householdID'. For unit-level model, this should be the variable name for cluster unit.

`pc.u` hyperparameter U for the PC prior on precisions.

`pc.alpha` hyperparameter alpha for the PC prior on precisions.

`pc.u.phi` hyperparameter U for the PC prior on the mixture probability phi in BYM2 model.

`pc.alpha.phi` hyperparameter alpha for the PC prior on the mixture probability phi in BYM2 model.

`CI` the desired posterior credible interval to calculate

`formula` a string of user-specified random effects model to be used in the INLA call

`timeVar` The variable indicating time period. If set to NULL then the temporal model and space-time interaction model are ignored.

`time.model` the model for temporal trends and interactions. It can be either "rw1" or "rw2".

`include_time_unstruct`
[Deprecated] The argument has been renamed into `include.time.unstruct`.

`include.time.unstruct`
Indicator whether to include the temporal unstructured effects (i.e., shocks) in the smoothed estimates from cluster-level model. The argument only applies to the unit-level models. Default is FALSE which excludes all unstructured temporal components. If set to TRUE all the unstructured temporal random effects will be included.

`type.st` can take values 0 (no interaction), or 1 to 4, corresponding to the type I to IV space-time interaction.

`direct.est` data frame of direct estimates, with column names of response and region specified by `responseVar`, `regionVar`, and `timeVar`. When `direct.est` is specified, it overwrites the `data` input.

`direct.est.var` the column name corresponding to the variance of direct estimates.

`is.unit.level` logical indicator of whether unit-level model is fitted instead of area-level model.

`is.agg` logical indicator of whether the input is at the aggregated counts by cluster. Only used for unit-level model and binary response variable.

`strataVar.within`
the variable specifying within region stratification variable. This is only used for the unit-level model.

`totalVar` the variable specifying total observations in `counts`. This is only used for the unit-level model when `counts` is specified.

`weight.strata` a data frame with one column corresponding to `regionVar`, and columns specifying proportion of each strata for each region. This argument specifies the weights for strata-specific estimates. This is only used for the unit-level model.

nsim	number of posterior draws to take. This is only used for the unit-level model when weight.strata is provided.
save.draws	logical indicator of whether to save the full posterior draws.
smooth	logical indicator of whether to perform smoothing. If set to FALSE, a data frame of direct estimate is returned. Only used when is.unit.level is FALSE.
...	additional arguments passed to svydesign function.

Details

The function smoothSurvey replaces the previous function name fitGeneric (before version 1.0.0).

Value

HT	Direct estimates
smooth	Smoothed direct estimates
fit	a fitted INLA object
CI	input argument
Amat	input argument
response.type	input argument
formula	INLA formula

Author(s)

Zehang Richard Li

See Also

getDirectList, smoothDirect

Examples

```
## Not run:
##
## 1. Area-level model with binary response
##

data(DemoData2)
data(DemoMap2)
fit0 <- smoothSurvey(data=DemoData2,
Amat=DemoMap2$Amat, response.type="binary",
responseVar="tobacco.use", strataVar="strata",
weightVar="weights", regionVar="region",
clusterVar = "~clustid+id", CI = 0.95)
summary(fit0)

# if only direct estimates without smoothing is of interest
fit0.dir <- smoothSurvey(data=DemoData2,
Amat=DemoMap2$Amat, response.type="binary",
```

```
responseVar="tobacco.use", strataVar="strata",
weightVar="weights", regionVar="region",
clusterVar = "~clustid+id", CI = 0.95, smooth = FALSE)

# posterior draws can be returned with save.draws = TRUE
fit0.draws <- smoothSurvey(data=DemoData2,
Amat=DemoMap2$Amat, response.type="binary",
responseVar="tobacco.use", strataVar="strata",
weightVar="weights", regionVar="region",
clusterVar = "~clustid+id", CI = 0.95, save.draws = TRUE)
# notice the posterior draws are on the latent scale
head(fit0.draws$draws.est[, 1:10])

# Example with region-level covariates
 Xmat <- aggregate(age~region, data = DemoData2,
FUN = function(x) mean(x))
 fit1 <- smoothSurvey(data=DemoData2,
  Amat=DemoMap2$Amat, response.type="binary",
  X = Xmat,
  responseVar="tobacco.use", strataVar="strata",
  weightVar="weights", regionVar="region",
  clusterVar = "~clustid+id", CI = 0.95)

# Example with using only direct estimates as input instead of the full data
direct <- fit0$direct[, c("region", "direct.est", "direct.var")]
fit2 <- smoothSurvey(data=NULL, direct.est = direct,
                     Amat=DemoMap2$Amat, regionVar="region",
                     responseVar="direct.est", direct.est.var = "direct.var",
                     response.type = "binary")
# Check it is the same as fit0
plot(fit2$smooth$mean, fit0$smooth$mean)

# Example with using only direct estimates as input,
#   and after transformation into a Gaussian smoothing model
# Notice: the output are on the same scale as the input
#   and in this case, the logit estimates.
direct.logit <- fit0$direct[, c("region", "direct.logit.est", "direct.logit.var")]
fit3 <- smoothSurvey(data=NULL, direct.est = direct.logit,
                Amat=DemoMap2$Amat, regionVar="region",
                responseVar="direct.logit.est", direct.est.var = "direct.logit.var",
                response.type = "gaussian")
# Check it is the same as fit0
plot(fit3$smooth$mean, fit0$smooth$logit.mean)

# Example with non-spatial smoothing using IID random effects
fit4 <- smoothSurvey(data=DemoData2, response.type="binary",
       responseVar="tobacco.use", strataVar="strata",
       weightVar="weights", regionVar="region",
       clusterVar = "~clustid+id", CI = 0.95)

# Example with missing regions in the raw input
DemoData2.sub <- subset(DemoData2, region != "central")
fit.without.central <- smoothSurvey(data=DemoData2.sub,
```

```
                        Amat=NULL, response.type="binary",
                        responseVar="tobacco.use", strataVar="strata",
                        weightVar="weights", regionVar="region",
                        clusterVar = "~clustid+id", CI = 0.95)
fit.without.central$direct
fit.without.central$smooth

fit.with.central <- smoothSurvey(data=DemoData2.sub,
                        Amat=NULL, region.list = unique(DemoData2$region),
                        response.type="binary",
                        responseVar="tobacco.use", strataVar="strata",
                        weightVar="weights", regionVar="region",
                        clusterVar = "~clustid+id", CI = 0.95)
fit.with.central$direct
fit.with.central$smooth

# Using the formula argument, further customizations can be added to the
#  model fitted. For example, we can fit the Fay-Harriot model with
#  IID effect instead of the BYM2 random effect as follows.
#  The "region.struct" and "hyperpc1" are picked to match internal object
#  names. Other object names can be inspected from the source of smoothSurvey.
fit5 <- smoothSurvey(data=DemoData2,
       Amat=DemoMap2$Amat, response.type="binary",
       formula = "f(region.struct, model = 'iid', hyper = hyperpc1)",
       pc.u = 1, pc.alpha = 0.01,
       responseVar="tobacco.use", strataVar="strata",
       weightVar="weights", regionVar="region",
       clusterVar = "~clustid+id", CI = 0.95)
# Check it is the same as fit4, notice the region order may be different
regions <- fit5$smooth$region
plot(fit4$smooth[match(regions, fit4$smooth$region),]$logit.mean, fit5$smooth$logit.mean)

##
## 2. Unit-level model with binary response
##

# For unit-level models, we need to create stratification variable within regions
data <- DemoData2
data$urbanicity <- "rural"
data$urbanicity[grep("urban", data$strata)] <- "urban"

# Beta-binomial likelihood is used in this model
fit6 <- smoothSurvey(data=data,
  Amat=DemoMap2$Amat, response.type="binary",
  X = Xmat, is.unit.level = TRUE,
  responseVar="tobacco.use", strataVar.within = "urbanicity",
  regionVar="region", clusterVar = "clustid", CI = 0.95)

# We may use aggregated PSU-level counts as input as well
#    in the case of modeling a binary outcome
data.agg <- aggregate(tobacco.use~region + urbanicity + clustid,
                      data = data, FUN = sum)
data.agg.total <- aggregate(tobacco.use~region + urbanicity + clustid,
```

```
                      data = data, FUN = length)
colnames(data.agg.total)[4] <- "total"
data.agg <- merge(data.agg, data.agg.total)
head(data.agg)

fit7 <- smoothSurvey(data=data.agg,
  Amat=DemoMap2$Amat, response.type="binary",
  X = Xmat, is.unit.level = TRUE, is.agg = TRUE,
  responseVar = "tobacco.use", strataVar.within = "urbanicity",
  totalVar = "total", regionVar="region", clusterVar = "clustid", CI = 0.95)

# Check it is the same as fit6
plot(fit6$smooth$mean, fit7$smooth$mean)

##
## 3. Area-level model with continuous response
##

# The smoothing model is the same as area-level model with binary response
#  the continuous direct estimates are smoothed instead of
#  their logit-transformed versions for binary response.
fit8 <- smoothSurvey(data=DemoData2, Amat=DemoMap2$Amat,
       response.type="gaussian", responseVar="age", strataVar="strata",
       weightVar="weights", regionVar="region",
       pc.u.phi = 0.5, pc.alpha.phi = 0.5,
       clusterVar = "~clustid+id", CI = 0.95)

##
## 4. Unit-level model with continuous response
##    (or nested error models)

# The unit-level model assumes for each of the i-th unit,
#    Y_{i} ~ intercept + region_effect + IID_i
#    where IID_i is the error term specific to i-th unit

# When more than one level of cluster sampling is carried out,
#   they are ignored here. Only the input unit is considered.
#   So here we do not need to specify clusterVar any more.
fit9 <- smoothSurvey(data= data,
  Amat=DemoMap2$Amat, response.type="gaussian",
  is.unit.level = TRUE, responseVar="age", strataVar.within = NULL,
  regionVar="region", clusterVar = NULL, CI = 0.95)

# To compare, we may also model PSU-level responses. As an illustration,
data.median <- aggregate(age~region + urbanicity + clustid,
                      data = data, FUN = median)

fit10 <- smoothSurvey(data= data.median,
  Amat=DemoMap2$Amat, response.type="gaussian",
  is.unit.level = TRUE, responseVar="age", strataVar.within = NULL,
  regionVar="region", clusterVar = "clustid", CI = 0.95)
```

```
# To further incorporate within-area stratification

fit11 <- smoothSurvey(data = data,
  Amat = DemoMap2$Amat, response.type = "gaussian",
  is.unit.level = TRUE, responseVar="age", strataVar.within = "urbanicity",
  regionVar = "region", clusterVar = NULL, CI = 0.95)

# Notice the usual output is now stratified within each region
# The aggregated estimates require strata proportions for each region
# For illustration, we set strata population proportions below
prop <- data.frame(region = unique(data$region),
                           urban = 0.3,
                           rural = 0.7)
fit12 <- smoothSurvey(data=data,
  Amat=DemoMap2$Amat, response.type="gaussian",
  is.unit.level = TRUE, responseVar="age", strataVar.within = "urbanicity",
  regionVar="region", clusterVar = NULL, CI = 0.95,
  weight.strata = prop)

# aggregated outcome
head(fit12$smooth.overall)

# Compare aggregated outcome with direct aggregating posterior means.
# There could be small differences if only 1000 posterior draws are taken.
est.urb <- subset(fit11$smooth, strata == "urban")
est.rural <- subset(fit11$smooth, strata == "rural")
est.mean.post <- est.urb$mean * 0.3 + est.rural$mean * 0.7
plot(fit12$smooth.overall$mean, est.mean.post)


##
## 6. Unit-level model with continuous response and unit-level covariate
##

# For area-level prediction, area-level covariate mean needs to be
#   specified in X argument. And unit-level covariate names are specified
#   in X.unit argument.

set.seed(1)
sim <- data.frame(region = rep(c(1, 2, 3, 4), 1000),
                  X1 = rnorm(4000), X2 = rnorm(4000))
Xmean <- aggregate(.~region, data = sim, FUN = sum)
sim$Y <- rnorm(4000, mean = sim$X1 + 0.3 * sim$X2 + sim$region)
samp <- sim[sample(1:4000, 20), ]
fit.sim <- smoothSurvey(data=samp ,
                 X.unit = c("X1", "X2"),
                 X = Xmean, Amat=NULL, response.type="gaussian",
                 is.unit.level = TRUE, responseVar="Y", regionVar = "region",
                 pc.u = 1, pc.alpha = 0.01, CI = 0.95)


## End(Not run)
```

`smoothUnit` *Smooth via basic unit level model*

Description

Generates small area estimates by smoothing direct estimates using a basic unit level model

Usage

```
smoothUnit(
  formula,
  domain,
  design,
  family = c("gaussian", "binomial")[1],
  X.pop = NULL,
  adj.mat = NULL,
  domain.size = NULL,
  pc.u = 1,
  pc.alpha = 0.01,
  pc.u.phi = 0.5,
  pc.alpha.phi = 2/3,
  level = 0.95,
  n.sample = 250,
  return.samples = F,
  X.pop.weights = NULL
)
```

Arguments

`formula`	An object of class 'formula' describing the model to be fitted.
`domain`	One-sided formula specifying factors containing domain labels
`design`	An object of class "svydesign" containing the data for the model
`family`	of the response variable, currently supports 'binomial' (default with logit link function) or 'gaussian'.
`X.pop`	Data frame of population unit-level covariates. One of the column name needs to match the domain specified, in order to be linked to the data input. Currently only supporting time-invariant covariates.
`adj.mat`	Adjacency matrix with rownames matching the domain labels. If set to NULL, the IID spatial effect will be used.
`domain.size`	Data frame of domain sizes. One of the column names needs to match the name of the domain variable, in order to be linked to the data input and there must be a column names 'size' containing domain sizes. The default option is no transformation, but logit and log are implemented.
`pc.u`	Hyperparameter U for the PC prior on precisions. See the INLA documentation for more details on the parameterization.

pc.alpha	Hyperparameter alpha for the PC prior on precisions.
pc.u.phi	Hyperparameter U for the PC prior on the mixture probability phi in BYM2 model.
pc.alpha.phi	Hyperparameter alpha for the PC prior on the mixture probability phi in BYM2 model.
level	The specified level for the posterior credible intervals
n.sample	Number of draws from posterior used to compute summaries
return.samples	If TRUE, return matrix of posterior samples of area level quantities
X.pop.weights	Optional vector of weights to use when aggregating unit level predictions

Value

A svysae object

Examples

```
## Not run:
data(DemoData2)
data(DemoMap2)
library(survey)
des0 <- svydesign(ids = ~clustid+id, strata = ~strata,
                  weights = ~weights, data = DemoData2, nest = TRUE)

# EXAMPLE 1: Continuous response model
cts.res <- smoothUnit(formula = tobacco.use ~ 1,
                      domain = ~region,
                      design = des0, X.pop = DemoData2)

# EXAMPLE 2: Binary response model
bin.res <- smoothUnit(formula = tobacco.use ~ 1,
                      family = "binomial",
                      domain = ~region,
                      design = des0, X.pop = DemoData2)

## End(Not run)
```

---

st.new *New Type I to IV space time interaction models for m-year to period random effects*

Description

New Type I to IV space time interaction models for m-year to period random effects

Usage

```
st.new(
  cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
  theta = NULL
)
```

Arguments

cmd	list of model components
theta	log precision

---

st.new.pc *New Type I to IV space time interaction models for m-year to period random effects*

Description

New Type I to IV space time interaction models for m-year to period random effects

Usage

```
st.new.pc(
  cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
  theta = NULL
)
```

Arguments

cmd	list of model components
theta	log precision

---

summary.SUMMERmodel *Summary method for the smoothing models.*

Description

This function is the summary method for class SUMMERmodel.

Usage

```
## S3 method for class 'SUMMERmodel'
summary(object, ...)
```

Arguments

object	output from smoothDirect or smoothCluster
...	not used

Author(s)

Zehang Li

See Also

summary.SUMMERmodel

Examples

```
## Not run:
  library(SUMMER)
  library(dplyr)
  data(DemoData)

  # Smooth Direct Model
  years <- levels(DemoData[[1]]$time)
  # obtain direct estimates
  data_multi <- getDirectList(births = DemoData, years = years,
  regionVar = "region",  timeVar = "time", clusterVar = "~clustid+id",
  ageVar = "age", weightsVar = "weights", geo.recode = NULL)
  data <- aggregateSurvey(data_multi)

  years.all <- c(years, "15-19")
  fit <- smoothDirect(data = data, Amat = NULL,
  year.label = years.all, year.range = c(1985, 2019),
  time.model = 'rw2', is.yearly=FALSE, m = 5)
  summary(fit)

  # Cluster-level Model
  counts.all <- NULL
  for(i in 1:length(DemoData)){
  counts <- getCounts(DemoData[[i]][, c("clustid", "time", "age", "died",
                                        "region", "strata")],
           variables = 'died', by = c("age", "clustid", "region",
                                      "time", "strata"))
  counts <- counts %>% mutate(cluster = clustid, years = time, Y=died)
  counts$strata <- gsub(".*\\.","",counts$strata)
  counts$survey <- names(DemoData)[i]
  counts.all <- rbind(counts.all, counts)
  }

  # fit cluster-level model on the periods
  periods <- levels(DemoData[[1]]$time)
  fit <- smoothCluster(data = counts.all,
     Amat = DemoMap$Amat,
     time.model = "rw2",
```

```
    st.time.model = "rw1",
    strata.time.effect =  TRUE,
    survey.effect = TRUE,
    family = "betabinomial",
    year.label = c(periods, "15-19"))
  summary(fit)

## End(Not run)
```

---

summary.SUMMERmodel.svy *Summary method for the smoothing model and output from* smoothSurvey.

Description

This function is the summary method for class SUMMERmodel.svy.

Usage

```
## S3 method for class 'SUMMERmodel.svy'
summary(object, ...)
```

Arguments

object	output from smoothSurvey
...	not used

Author(s)

Zehang Li

See Also

summary.SUMMERmodel.svy

Examples

```
## Not run:
data(DemoData2)
data(DemoMap2)
fit0 <- smoothSurvey(data=DemoData2,
Amat=DemoMap2$Amat, responseType="binary",
responseVar="tobacco.use", strataVar="strata",
weightVar="weights", regionVar="region",
clusterVar = "~clustid+id", CI = 0.95)
summary(fit0)

## End(Not run)
```

summary.SUMMERprojlist *Summary method for the combined projection output. This function is the print method for class* SUMMERprojlist.

Description

Summary method for the combined projection output. This function is the print method for class SUMMERprojlist.

Usage

```
## S3 method for class 'SUMMERprojlist'
summary(object, ...)
```

Arguments

object	output from getSmoothed
...	not used

Author(s)

Zehang Li

Examples

```
## Not run:
 library(SUMMER)
 library(dplyr)
 data(DemoData)
 # Create dataset of counts
 counts.all <- NULL
 for(i in 1:length(DemoData)){
 counts <- getCounts(DemoData[[i]][, c("clustid", "time", "age", "died",
                                       "region", "strata")],
          variables = 'died', by = c("age", "clustid", "region",
                                       "time", "strata"))
 counts <- counts %>% mutate(cluster = clustid, years = time, Y=died)
 counts$strata <- gsub(".*\\.","",counts$strata)
 counts$survey <- names(DemoData)[i]
 counts.all <- rbind(counts.all, counts)
 }

 # fit cluster-level model on the periods
 periods <- levels(DemoData[[1]]$time)
 fit <- smoothCluster(data = counts.all,
    Amat = DemoMap$Amat,
    time.model = "rw2",
    st.time.model = "rw1",
```

```
    strata.time.effect =  TRUE,
    survey.effect = TRUE,
    family = "betabinomial",
    year.label = c(periods, "15-19"))
 summary(fit)
 est <- getSmoothed(fit, nsim = 1000)

## End(Not run)
```

`tcpPlot` *Discrete-color maps based on the True Classification Probabilities*

Description

Discrete-color maps based on the True Classification Probabilities

Usage

```
tcpPlot(
  draws,
  geo,
  by.geo = NULL,
  year.plot = NULL,
  year_plot = deprecated(),
  ncol = 4,
  per1000 = FALSE,
  thresholds = NULL,
  intervals = 3,
  size.title = 0.7,
  legend.label = NULL,
  border = "gray20",
  size = 0.5
)
```

Arguments

`draws`	a posterior draw object from `getSmoothed`
`geo`	SpatialPolygonsDataFrame object for the map
`by.geo`	variable name specifying region names in geo
`year.plot`	vector of year string vector to be plotted.
`year_plot`	**[Deprecated]** replaced by year.plot
`ncol`	number of columns in the output figure.
`per1000`	logical indicator to multiply results by 1000.
`thresholds`	a vector of thresholds (on the mortality scale) defining the discrete color scale of the maps.

intervals	number of quantile intervals defining the discrete color scale of the maps. Required when thresholds are not specified.
size.title	a numerical value giving the amount by which the plot title should be magnified relative to the default.
legend.label	Label for the color legend.
border	color of the border
size	size of the border

Value

a list of True Classification Probability (TCP) tables, a list of individual spplot maps, and a gridded array of all maps.

Author(s)

Tracy Qi Dong, Zehang Richard Li

References

Tracy Qi Dong, and Jon Wakefield. (2020) *Modeling and presentation of vaccination coverage estimates using data from household surveys.* arXiv preprint arXiv:2004.03127.

Examples

```
## Not run:
library(dplyr)
data(DemoData)
# Create dataset of counts, unstratified
counts.all <- NULL
for(i in 1:length(DemoData)){
  counts <- getCounts(DemoData[[i]][, c("clustid", "time", "age", "died",
                                        "region")],
            variables = 'died', by = c("age", "clustid", "region",
                                       "time"))
  counts <- counts %>% mutate(cluster = clustid, years = time, Y=died)
  counts$strata <- NA
  counts$survey <- names(DemoData)[i]
  counts.all <- rbind(counts.all, counts)
}

# fit cluster-level model on the periods
periods <- levels(DemoData[[1]]$time)
fit <- smoothCluster(data = counts.all,
      Amat = DemoMap$Amat,
      time.model = "rw2",
      st.time.model = "rw1",
      strata.time.effect =  TRUE,
      survey.effect = TRUE,
      family = "betabinomial",
      year.label = c(periods, "15-19"))
```

```
est <- getSmoothed(fit, nsim = 1000, save.draws=TRUE)

tcp <- tcpPlot(est, DemoMap$geo, by.geo = "REGNAME", interval = 3, year.plot = periods)
tcp$g

## End(Not run)
```

Index