ARTnet: Parameterization for HIV/STI Models

Shawnee Anderson and Samuel M. Jenness

2019-11-26

## Load the Package

library("ARTnet")

## Geographic Stratification

This new input tells build\_epistats that the geographical level of interest is the city level and that the geographical region of interest is Atlanta. There are five levels of geographical subsetting available in ARTnet: city, state, census division and geographic region. These are called by geog.lvl = city, state, div, and reg respectively in the build\_epistats function call. An individual who would like to build a network structure based on, say, Georgia, would then use:

epistats2 <- build\_epistats(geog.lvl = "state", geog.cat = "GA")

Similarly, if one wishes to use the other levels of geographic stratification they can do so as above:

epistats2 <- build\_epistats(geog.lvl = "division", geog.cat = "5")  
epistats3 <- build\_epistats(geog.lvl = "region", geog.cat = "3")

If no geographic stratification is specified, geog.lvl = NULL, and no geopgraphic stratification is done.

epistats4 <- build\_epistats(geog.lvl = NULL)

## Race Stratification

A further change made to ARTnet is the ability for users to choose whether or not to add race into network estimation. Previously, this was “on”" by default; however now, with the race = option in build\_epistats, users are able to choose between TRUE or FALSE denoting race and no-race stratification:

#Racial stratification  
epistats4 <- build\_epistats(geog.lvl = "state", geog.cat = "WA", race = TRUE)  
  
#No racial stratification  
epistats5 <- build\_epistats(geog.lvl = "state", geog.cat = "GA", race = FALSE)

Racial stratification is set to true by default.

## Age Stratification

A further addition to the ARTnet workflow is allowing for age group subsetting and arbitrary setting of age groups. Currently ARTnet uses the full range of ages within ARTnetData and age groups are set at 0, 24, 34, 44, 54, 64, 100. This is retained as the default behavior behavior of ARTnet with new variables age.limits in build\_epistats and age.breaks allowing for age range and group specification respectively:

* age.limits: specifies the range of values to subset ARTnetData by.
* age.breaks: explicitly specifies the age breaks for age categories. If NULL defaults to 5 equally spaced categories.

#Set age limits  
epistats6 <- build\_epistats(geog.lvl = "state", geog.cat = "GA",   
 race = TRUE, age.limits = c(15, 65),  
 age.breaks = c(20, 30, 40, 50, 60))  
  
#Specify age categories: (0-20], (20, 30], . . . (60, 100]  
netparams1 <- build\_netparams(epistats = epistats6, smooth.main.dur = TRUE)

## Full EpiModel Workflow

Below is the updated workflow for generating network objects for use in constructing full network parameters/statistics for use in EpiModel:

#1. Epistats: Specify geographic features, as well as race stratification and total age range  
#under consideration  
epistats <- build\_epistats(geog.lvl = "city", geog.cat = "Atlanta", race = TRUE,   
 age.limits = c(30, 50), age.breaks = c(35, 45))  
  
#2. Netparams: Specify age categories if needed, or let ARTnet determine age categories by number of   
#categories desired  
netparams <- build\_netparams(epistats = epistats, smooth.main.dur = TRUE)  
  
#3. Netstats: Finalize network setup   
netstats <- build\_netstats(epistats, netparams, expect.mort = 0.0005,   
 network.size = 1000, edges.avg =TRUE)  
  
#4. Initialize network using `netstats` object from previous step.  
num <- netstats$demog$num  
nw <- network::network.initialize(num, directed = FALSE)  
  
attr.names <- names(netstats$attr)  
attr.values <- netstats$attr  
nw <- network::set.vertex.attribute(nw, attr.names, attr.values)  
nw\_main <- nw\_casl <- nw\_inst <- nw  
  
# 5. Main Model  
  
#Formula:   
model\_main <- ~edges +  
 nodematch("age.grp", diff = TRUE) +  
 nodefactor("age.grp", base = 1) +  
 nodematch("race", diff = FALSE) + #race = TRUE; omit if FALSE  
 nodefactor("race", base = 1) +  
 nodefactor("deg.casl", base = 1) +  
 concurrent +  
 degrange(from = 3) +  
 nodematch("role.class", diff = TRUE, keep = 1:2)  
  
# Target Stats  
netstats\_main <- c(  
 edges = netstats$main$edges,  
 nodematch\_age.grp = netstats$main$nodematch\_age.grp,  
 nodefactor\_age.grp = netstats$main$nodefactor\_age.grp[-1],  
 nodematch\_race = netstats$main$nodematch\_race\_diffF, #If race = FALSE, value will be NULL  
 nodefactor\_race = netstats$main$nodefactor\_race[-1],  
 nodefactor\_deg.casl = netstats$main$nodefactor\_deg.casl[-1],  
 concurrent = netstats$main$concurrent,  
 degrange = 0,  
 nodematch\_role.class = c(0, 0)  
)  
cbind(netstats\_main)  
netstats\_main <- unname(netstats\_main)  
  
# Fit model  
fit\_main <- netest(nw\_main,  
 formation = model\_main,  
 target.stats = netstats\_main,  
 coef.diss = netstats$main$diss.byage,  
 set.control.ergm = control.ergm(MCMLE.maxit = 500,  
 SAN.maxit = 3,  
 SAN.nsteps.times = 3),  
 verbose = FALSE)  
  
# 6. Casual Model  
  
# Formula  
model\_casl <- ~edges +  
 nodematch("age.grp", diff = TRUE) +  
 nodefactor("age.grp", base = c(1,5)) +  
 nodefactor("deg.main", base = 3) +  
 concurrent +  
 degrange(from = 4) +  
 nodematch("role.class", diff = TRUE, keep = 1:2) +  
 #If race = TRUE:  
 nodematch("race", diff = FALSE) +  
 nodefactor("race", base = 1)  
  
# Target Stats  
netstats\_casl <- c(  
 edges = netstats$casl$edges,  
 nodematch\_age.grp = netstats$casl$nodematch\_age.grp,  
 nodefactor\_age.grp = netstats$casl$nodefactor\_age.grp[-c(1,5)],  
 nodefactor\_deg.main = netstats$casl$nodefactor\_deg.main[-3],  
 concurrent = netstats$casl$concurrent,  
 degrange = 0,  
 nodematch\_role.class = c(0, 0),  
 #If race = TRUE:  
 nodematch\_race = netstats$casl$nodematch\_race\_diffF,   
 nodefactor\_race = netstats$casl$nodefactor\_race[-1]  
)  
cbind(netstats\_casl)  
netstats\_casl <- unname(netstats\_casl)  
  
# Fit model  
fit\_casl <- netest(nw\_casl,  
 formation = model\_casl,  
 target.stats = netstats\_casl,  
 coef.diss = netstats$casl$diss.byage,  
 set.control.ergm = control.ergm(MCMLE.maxit = 500,  
 SAN.maxit = 3,  
 SAN.nsteps.times = 3),  
 verbose = FALSE)  
  
# 7. One-Off Model  
  
# Formula  
model\_inst <- ~edges +  
 nodematch("age.grp", diff = FALSE) +  
 nodefactor("age.grp", base = 1) +  
 nodefactor("risk.grp", base = 5) +  
 nodefactor("deg.tot", base = 1) +  
 nodematch("role.class", diff = TRUE, keep = 1:2) +  
 #If race = TRUE  
 nodematch("race", diff = FALSE) +  
 nodefactor("race", base = 1)   
  
# Target Stats  
netstats\_inst <- c(  
 edges = netstats$inst$edges,  
 nodematch\_age.grp = sum(netstats$inst$nodematch\_age.grp),  
 nodefactor\_age.grp = netstats$inst$nodefactor\_age.grp[-1],  
 nodefactor\_risk.grp = netstats$inst$nodefactor\_risk.grp[-5],  
 nodefactor\_deg.tot = netstats$inst$nodefactor\_deg.tot[-1],  
 nodematch\_role.class = c(0, 0),  
 #If race = TRUE  
 nodematch\_race = netstats$inst$nodematch\_race\_diffF,  
 nodefactor\_race = netstats$inst$nodefactor\_race[-1]  
)  
cbind(netstats\_inst)  
netstats\_inst <- unname(netstats\_inst)  
  
# Fit model  
fit\_inst <- netest(nw\_inst,  
 formation = model\_inst,  
 target.stats = netstats\_inst,  
 coef.diss = dissolution\_coefs(~offset(edges), 1),  
 set.control.ergm = control.ergm(MCMLE.maxit = 500,  
 SAN.maxit = 3,  
 SAN.nsteps.times = 3),  
 verbose = FALSE)  
  
# 8. Save Data  
  
out <- list(fit\_main, fit\_casl, fit\_inst)  
  
# saveRDS(out, file = "netest.rda")