INA package - node number effect on speed

K. A. Garrett, University of Florida

Effects of increasing the number of nodes on function speed

This analysis addresses how the speed of processing for INA functions scales with the number of nodes in the networks. For two main functions, smartsurv and INAscene, the functions are implemented for a range of number of nodes and the speed is compared using the package tictoc.

Getting the INApreliminary package from GitHub

An updated version of INApreliminary is available on GitHub.

Note that if you do not already have the package devtools, you will need to install it once (using the command commented out below)

Then you will need to install INApreliminary from GitHub (command commented out below)

```
# use the following three commented commands if you do not yet have
# INApreliminary or need to update the version

#install.packages("devtools") # use this command if you do not yet have devtools installed

#library(devtools)

#devtools::install_github("GarrettLab/INApreliminary")

library(INApreliminary)

library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
library(tictoc)
```

The INA function smartsurv

Consider the application of smartsurv for adjacency matrices with numbers of nodes varying.

```r
# a function for testing the timing of smartsurv as a function of node number

sstest <- function(nnodes) {
  # generate a directed random network with nnodes nodes
  randnn <- sample_gnp(
  n = nnodes,
  p = 0.05,
  directed = T,
  loops = F
  )

  # find the value of nodes for sampling based on a deterministic model

  temp1 <- as_adjacency_matrix(randnn, sparse = F)
  diag(temp1) <- 1 # nodes invaded stay invaded

  # test the speed using tictoc (note that this has stoch=F so is faster than it
  # would be with stoch=T)

  tic()
  temp2 <- smartsurv(adjmat = temp1,
  stoch = F,
  nrealz = 1)
  toc()
  }
```

Here are some results for a basic laptop, approximating the time for 1 node by the time for 3 nodes, or 0.01

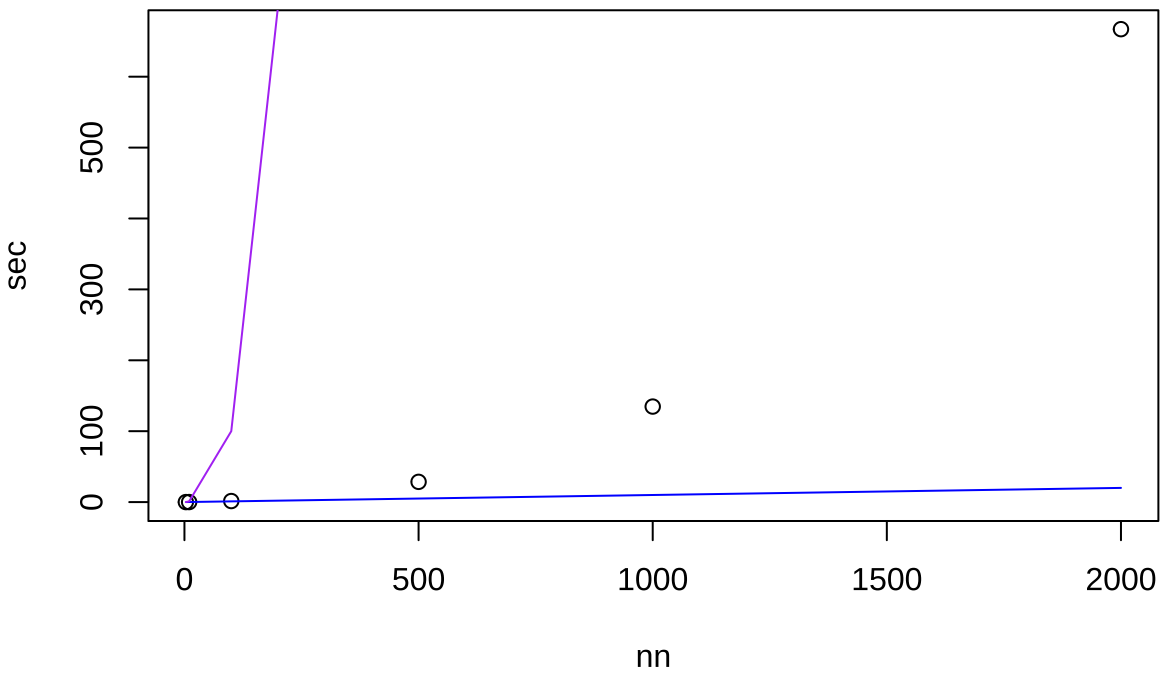
```r
nn <- c(3, 10, 100, 500, 1000, 2000) # the number of nodes evaluated

# the observed seconds required for completion on a basic laptop
sec <- c(0.01, 0.02, 1.48, 28.54, 134.73, 667.02)

plot(nn,sec)

# example for time to completion equal to the number of  nodes (scaled by 0.01
# seconds for 3 nodes)
lines(nn, (0.01 * nn), col='blue')

# example for time to completion equal to the number of  nodes squared (scaled
# by 0.01 seconds for 3 nodes)
lines(nn, (0.01 * nn^2), col='purple')
```



```
# lines(nn, (0.01 * nn^3), col='red') # example for time to completion equal to
# the number of nodes cubed (scaled by 0.01 seconds for 3 nodes)
```

For the smartsurv function, the time required to complete the function scales with the number of nodes at a rate greater than n (blue line) but less than n^2 (purple line). The smartsurv function evaluates multiple epidemics until a stopping point is reached indicating the 'end' of an epidemic. Epidemics starting at a single location take longer to 'end' the more nodes are being considered.

INAscene function

For the INAscene function, here's a similar example for one realization, for a case where networks are generated within INAscene.

When there are multiple realizations, time can be saved by using multiple cores.

Note that, when INAscene generates adjacency networks based on geographic positions, the size of the area being considered may also be a factor for speed. In this example, the size of the area being considered scales with the square root of the number of nodes.

```
# a function for testing the timing of smartsurv as a function of node number
# and the number of time steps considered

istest <- function(nnodes, ntimes) {
  # evaluate one realization using the INAscene function, testing the speed
  # using tictoc

  tic()

  j25 <-
  INAscene(
  nreals = 1,
  ntimesteps = ntimes,
```

```r
  doplot = F,
  readgeocoords = F,
  geocoords = NA,
  numnodes = nnodes,
  xrange = c(0, sqrt(nnodes)),
  yrange = c(0, sqrt(nnodes)),
  randgeo = T,
  readinitinfo = F,
  initinfo = NA,
  initinfo.norp = 'num',
  initinfo.n = 5,
  initinfo.p = NA,
  initinfo.dist = 'random',
  readinitbio = F,
  initbio = NA,
  initbio.norp = 'num',
  initbio.n = 5,
  initbio.p = NA,
  initbio.dist = 'random',
  readseam = F,
  seam = NA,
  seamdist = 'random',
  seamrandp = 0.1,
  seampla = NA,
  seamplb = NA,
  readbpam = F,
  bpam = NA,
  bpamdist = 'random',
  bpamrandp = 0.1,
  bpampla = NA,
  bpamplb = NA,
  readprobadoptvec = F,
  probadoptvec = NA,
  probadoptmean = 0.1,
  probadoptsd = 0.1,
  readprobestabvec = F,
  probestabvec = NA,
  probestabmean = 0.1,
  probestabsd = 0.1,
  maneffdir = 'decrease_estab',
  maneffmean = 0.5,
  maneffsd = 0.1,
  usethreshman = F,
  maneffthresh = NA,
  sampeffort = NA
  )

  toc()
  }
```

The following results are times from a basic laptop, using the function istest for 3 times steps with the number of nodes ranging from 5 to 5000.

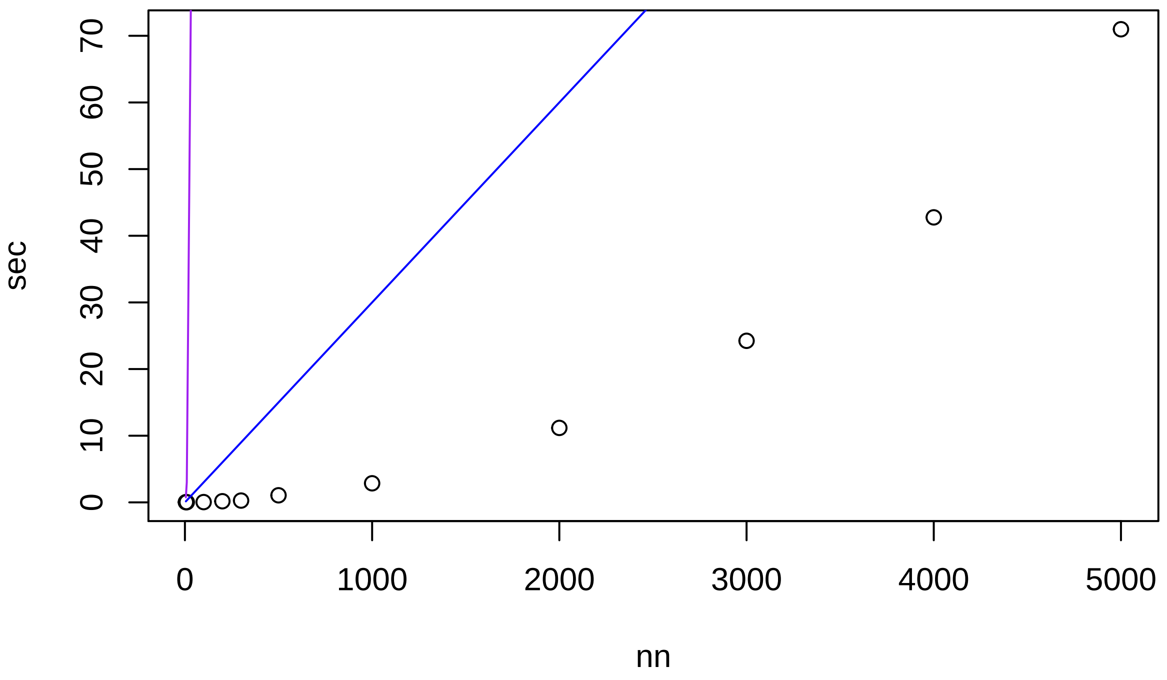
```r
# the number of nodes evaluated
nn <- c(5, 10, 100, 200, 300, 500, 1000, 2000, 3000, 4000, 5000)

# the observed seconds required for completion on a basic laptop
sec <- c(0.03, 0.03, 0.04, 0.17, 0.27, 1.06, 2.86, 11.17, 24.24, 42.76, 70.98)

plot(nn,sec)

# example for time to completion equal to the number of  nodes (scaled by 0.03
# seconds for 5 nodes)
lines(nn, (0.03 * nn), col='blue')

# example for time to completion equal to the number of  nodes squared (scaled
# by 0.03 seconds for 5 nodes)
lines(nn, (0.03 * nn^2), col='purple')
```



```r
# lines(nn, (0.03 * nn^3), col='red') # example for time to completion equal to
# the number of nodes cubed (scaled by 0.03 seconds for 5 nodes)
```

INAscene scales more efficiently than smartsurv, but users of INAscene may want to evaluate hundreds of realizations for each parameter combination, and potentially dozens of parameter combinations. Thus, for larger applications, spreading analyses across multiple cores, and ideally use of a supercomputer, could make extensive analyses more practical when results are needed quickly.

References

Csardi, G., and T. Nepusz. 2006. The igraph software package for complex network research. InterJournal, Complex Systems: 1695. http://igraph.org

Izrailev, S. 2014. tictoc: Functions for timing R scripts, as well as implementations of Stack and List structures. R package version 1.0. https://CRAN.R-project.org/package=tictoc

Wickham, H., J. Hester, and W. Chang. devtools: Tools to Make Developing R Packages Easier. 2020. R package version 2.3.0. https://CRAN.R-project.org/package=devtools