Social Network Analysis Home Assignment 4

Shupeyko Nikita

due date - 12.06.2016 23:59

Contents

Network Epidemics

SIR Model

You need to perform epidemic SIR model on different types of networks: Try different parameters for network generation:

```
gs = list()

# The number of vertices must be the same among the graphs,
# and so are edge counts/edge probabilities.
# The graph represented by Net.txt is the largest one.
gs$ptp = read.graph("Net.txt")
gs$ptp = as.undirected(gs$ptp)
gs$ptp$name = "Net"
ptp.n = vcount(gs$ptp)
ptp.m = ecount(gs$ptp)
ptp.avg.node.deg = 2*ptp.m / ptp.n
ptp.p = 2*ptp.m / (ptp.n*(ptp.n-1))

gs$ba = barabasi.game(n = ptp.n,
                      m = ptp.avg.node.deg,
                      directed = F)
gs$ba$name = "Barabasi"

gs$er = erdos.renyi.game(n = ptp.n,
                         p.or.m = ptp.m,
                         type = c("gnm"),
                         directed = F)
gs$er$name = "Erdos-Renyi"

gs$ws = watts.strogatz.game(dim = 1,
                            size = ptp.n,
                            nei = 3,
                            p = ptp.p)
gs$ws$name = "Watts-Strogatz"
```

Moreover, perform modeling on real peer-to-peer network here

Your goal is to perform a research on epidemics:

```
simulate.sir.on.graphs = function(graphs,
                                  beta,
                                  gamma,
                                  no.sim) {
  sim = list(results = lapply(graphs,
                              sir,
                              beta = beta,
                              gamma = gamma,
                              no.sim = no.sim),
             params =  list(beta = beta,
                            gamma = gamma,
                            no.sim = no.sim))
  return(sim)
}

# For the sake of consistency, the same number of trials
# was kept across all simulations.
sims = list(`1` = simulate.sir.on.graphs(gs, 3, 9, 250),
            `2` = simulate.sir.on.graphs(gs, 6, 6, 250),
            `3` = simulate.sir.on.graphs(gs, 9, 3, 250))
```

at least 3 different versions, for example:

- betta (4 6 8)
- gamma (8 6 2)
- niter (100 500 1000)

For some reason `beta` *and* `gamma` *parameters should not be set below 0 and 1. Looks like they are somehow normilized during simulation.*

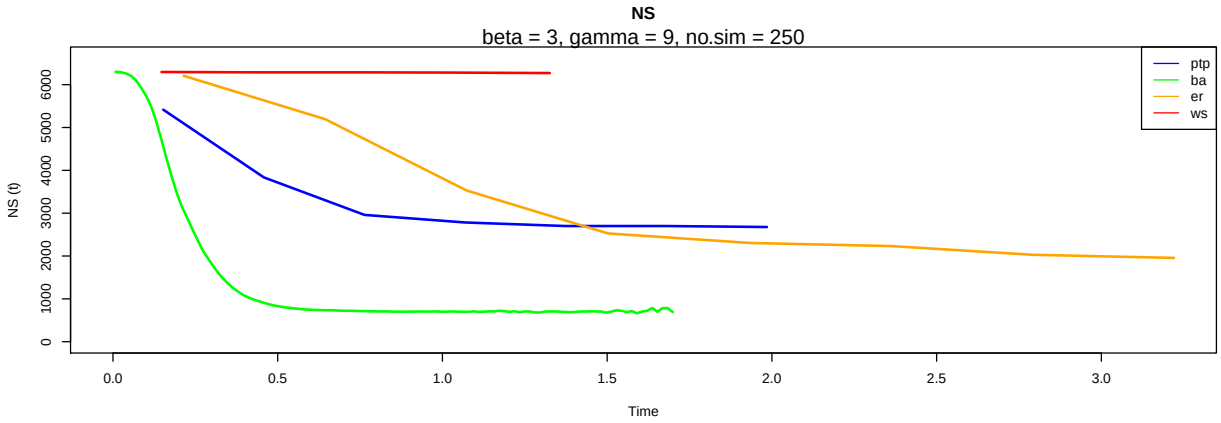
You need to plot three values on the graphics: Number of infected, number of suseprible, number of recovered - all depends on time. As a result of this task, you need to provide 12 plots (one for each network with 3 diferent parameters) with explanation.‘

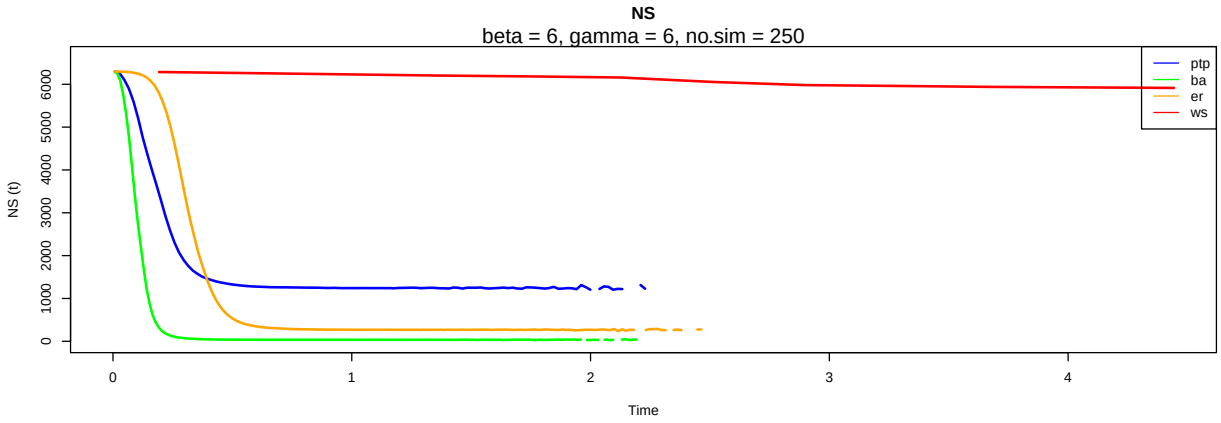
```
plot.simulation = function(simulation,
                           graph.colors,
                           kind = c("NS", "NI", "NR")) {
  if (length(simulation) == 0) {
    stop("simulation is NULL or empty!")
  }
  if (length(simulation$results) == 0) {
    stop("simulation$results is NULL or empty!")
  }
  if (length(simulation$params) == 0) {
    stop("simulation$params is NULL or empty!")
  }
  graph.names = names(simulation$results)
  if (length(graph.names) != length(graph.colors)) {
    stop("The number of graphs in simulation does not match the number of colors specified!")
  }
  if (!(kind %in% c("NS", "NI", "NR"))) {
    stop("Cannot plot anything of kind other than 'NS', 'NI', or 'NR'!")
  }

  x.max = max(sapply(sapply(simulation$results,
                            time_bins),
```

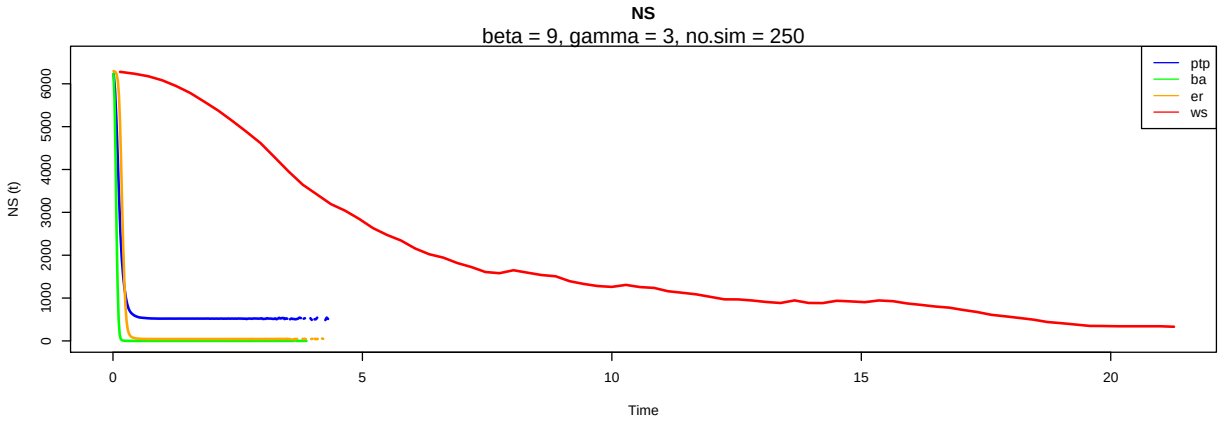
```
                    max))
  y.max = 1.05 * max(sapply(sapply(simulation$results,
                                   function(x) median(x)[[kind]]),
                            max,
                            na.rm = T))
  for (graph.name.i in seq_along(graph.names)) {
    graph.name = graph.names[graph.name.i]
    results.for.graph = simulation$results[[graph.name]]
    if (graph.name.i == 1) {
      plot(time_bins(results.for.graph),
           median(results.for.graph)[[kind]],
           type = "l",
           xlim = c(0, x.max),
           ylim = c(0, y.max),
           xlab = "Time",
           ylab = paste0(kind, " (t)"),
           lwd = 2,
           col = graph.colors[graph.name.i])
      title(kind)
      mtext(sprintf("beta = %i, gamma = %i, no.sim = %i",
                    simulation$params$beta,
                    simulation$params$gamma,
                    simulation$params$no.sim))
    } else {
      lines(time_bins(results.for.graph),
            median(results.for.graph)[[kind]],
            lwd = 2,
            col = graph.colors[graph.name.i])
    }
  }
  legend("topright",
         graph.names,
         col = graph.colors,
         lty = 1)
}


graph.colors = c("blue", "green", "orange", "red")
```

NS Comparison Charts

```
par(mfrow = c(3, 1))
for (sim.name in names(sims)) {
  plot.simulation(sims[[sim.name]], graph.colors, kind = "NS")
}
```







While `Net`, `Barabasi`, and `Erdos-Renyi` networks demonstrate similiar behavior (all three start decreasing rapidly almost immediately after the simulation is started, reaching an inverse plateau in a while), the `Watts-Strogatz` remains unreceptive to epidemy much longer (up to the second time unit) – it is a challenge

to figure out why.

First, let's take a look at a graph decriptives comparison table:

```
print.descriptives.for.graphs = function(graphs.list,
                                         include.density = T,
                                         include.diameter = T,
                                         include.clustering.coefficient = T,
                                         include.cluster.count = T) {
  tbl = NULL

  for(g.name in names(graphs.list)) {
    g = graphs.list[[g.name]]

    tbl.column = matrix()
    colnames(tbl.column) = g$name
    rownames(tbl.column) = c(NA)

    if (include.density) {
      g.dens = graph.density(g)
      tbl.column = rbind(tbl.column,
                         g.dens)
      rownames(tbl.column)[length(rownames(tbl.column))] = "Density"
    }
    if (include.diameter) {
      g.diam = diameter(g)
      tbl.column = rbind(tbl.column,
                         g.diam)
      rownames(tbl.column)[length(rownames(tbl.column))] = "Diameter"
    }
    if (include.clustering.coefficient) {
      g.cc = transitivity(g, type = "global")
      tbl.column = rbind(tbl.column,
                         g.cc)
      rownames(tbl.column)[length(rownames(tbl.column))] = "Clustering Coefficient"
    }
    if (include.cluster.count) {
      g.c.no = clusters(g)$no
      tbl.column = rbind(tbl.column,
                         g.c.no)
      rownames(tbl.column)[length(rownames(tbl.column))] = "Cluster Count"
    }

    if (is.null(tbl)) {
      tbl = tbl.column
    } else {
      tbl = cbind(tbl,
                  tbl.column)
    }
  }

  tbl = tbl[-1, ]
  tbl = xtable(tbl,
               caption = "Graph Comparison")
  digits(tbl) = c(0, rep(4, length(graphs.list)))
```

```
  align(tbl) = paste0("r",
                      paste0(rep("|c", length(graphs.list)),
                             collapse = ""))
  print(tbl,
        type = "latex",
        floating = T,
        include.rownames = T,
        latex.environments = "center",
        caption.placement = "top")
}


print.descriptives.for.graphs(gs)
```

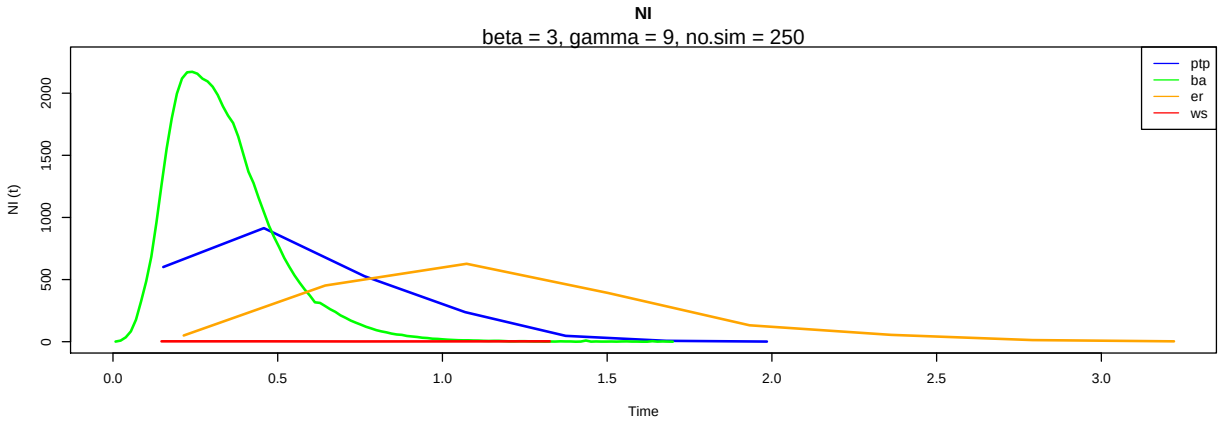
Table 1: Graph Comparison

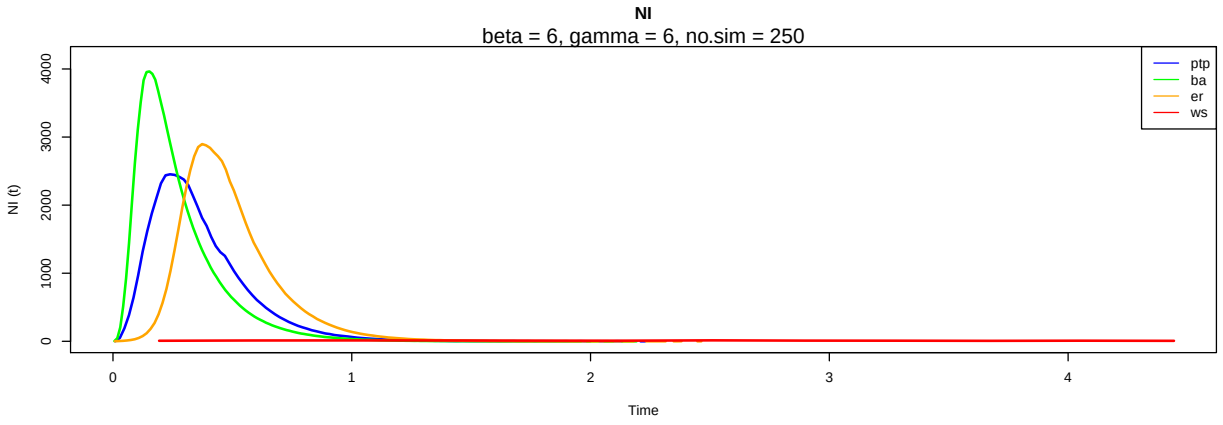
	Net	Barabasi	Erdos-Renyi	Watts-Strogatz
Density	0.0010	0.0019	0.0010	0.0010
Diameter	9.0000	5.0000	10.0000	168.0000
Clustering Coefficient	0.0207	0.0086	0.0011	0.5966
Cluster Count	2.0000	1.0000	8.0000	1.0000

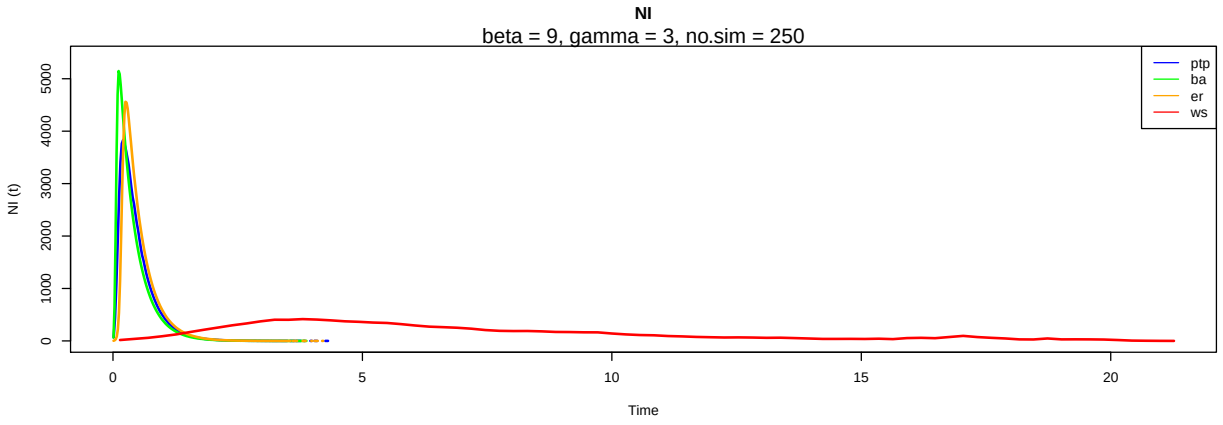
Clearly, the `Watts-Strogatz`'s diameter is outstanding, which justifies the long-running SIR simuluation: It is just the epidemy that needs a fair amount of time to spread.

NI Comparison Charts

```
par(mfrow = c(3, 1))
for (sim.name in names(sims)) {
  plot.simulation(sims[[sim.name]], graph.colors, kind = "NI")
}
```



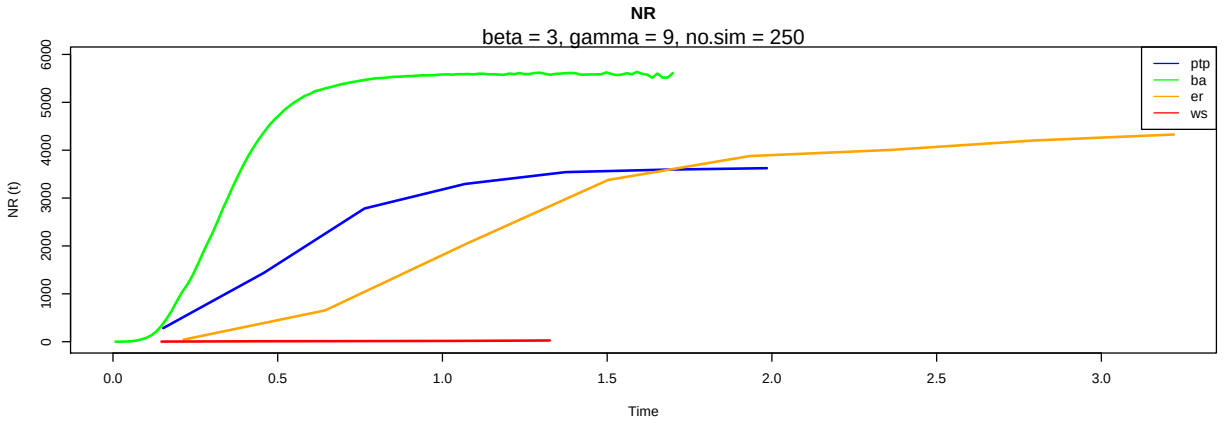


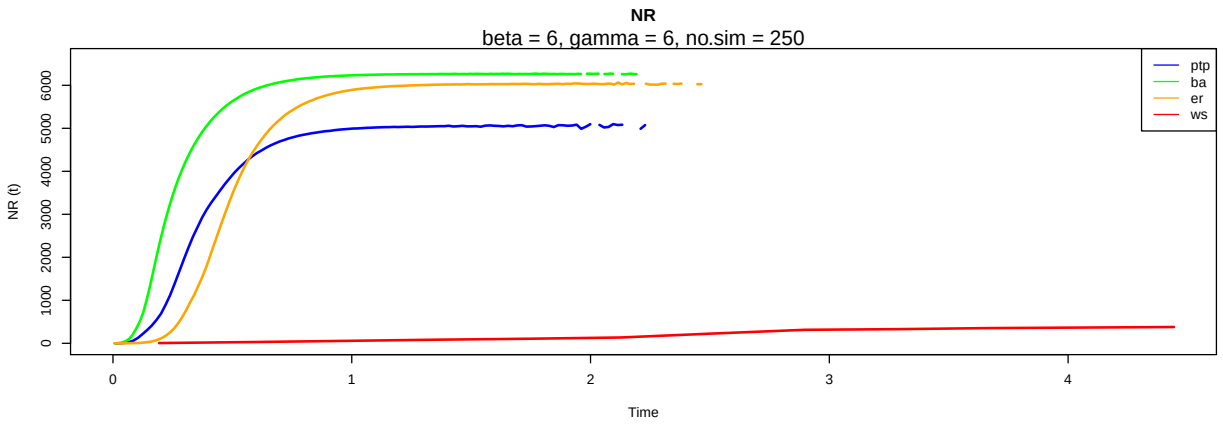


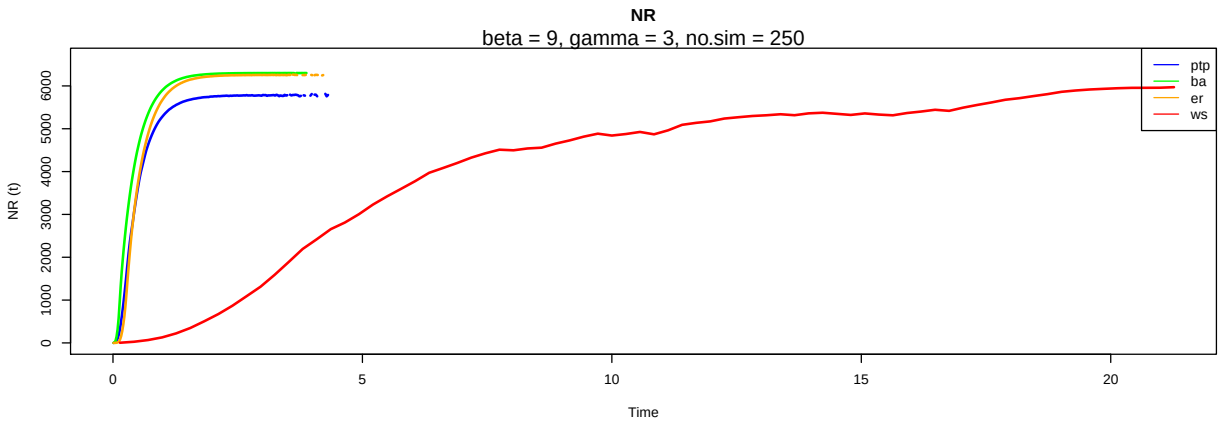
Except for the `Watts-Strogatz` series, other types of network expose similiar behaviour over time, each hitting its peak around the first time unit. By contrast, due to the `Watts-Strogatz`'s diameter infection spread is negligible (even for $\beta \gg \gamma$, its footprint is nowhere close to that in other models).

NR Comparison Charts

```
par(mfrow = c(3, 1))
for (sim.name in names(sims)) {
  plot.simulation(sims[[sim.name]], graph.colors, kind = "NR")
}
```







Once again, the good-old `Watts-Strogatz` stands out: for $\beta \gg \gamma$, the number of recovered actors is nearly 0; for $\beta = \gamma$, a small number of previously infected actors recover in a graduate manner; the last plot exposes a slight ununiformity in NR by the very end of simulation, which is likely justified by the respective networks

structure.