In Silico Experiments with relevent

Carter T. Butts

5/16/2023

Contents

Last updated 2023-05-16

This tutorial is a joint product of the Statnet Development Team:

Pavel N. Krivitsky (University of New South Wales)
Martina Morris (University of Washington)
Mark S. Handcock (University of California, Los Angeles)
Carter T. Butts (University of California, Irvine)
David R. Hunter (Penn State University)
Steven M. Goodreau (University of Washington)
Chad Klumb (University of Washington)
Skye Bender de-Moll (Oakland, CA)
Michał Bojanowski (Kozminski University, Poland)

The network modeling software demonstrated in this tutorial is authored by Carter Butts (`relevent`, `sna`).

The `statnet` Project

All `statnet` packages are open-source, written for the **R** computing environment, and published on CRAN. The source repositories are hosted on GitHub. Our website is statnet.org

- Need help? For general questions and comments, please email the statnet users group at statnet_help@uw.edu. You'll need to join the listserv if you're not already a member. You can do that here: statnet_help listserve.
- Found a bug in our software? Please let us know by filing an issue in the appropriate package GitHub repository, with a reproducible example.
- Want to request new functionality? We welcome suggestions – you can make a request by filing an issue on the appropriate package GitHub repository. The chances that this functionality will be developed

are substantially improved if the requests are accompanied by some proposed code (we are happy to review pull requests).

- For all other issues, please email us at contact@statnet.org.

Section 0. Introduction to the Tutorial

This tutorial provides a very basic introduction to *in silico* experiments with relational event data using `statnet` software. This online tutorial is also designed for self-study, with example code and self-contained data. We will be using a number of `statnet` tools, notably including:

- `relevent` – modeling and simulation for relational event models
- `sna` – tools for social network analysis

Additional background on the tools, modeling framework, and data used in this tutorial may be found in the references at the bottom of this document.

0.0 Prerequisites

This tutorial assumes basic familiarity with **R**, experience with network concepts, terminology and data, and familiarity with the general framework for statistical modeling and inference. It also assumes basic familiarity with relational event models (REMs). If you are not familiar with REMs, you may want to examine the `statnet` tutorial on relational event models first. (See the `statnet` web site for available tutorials.) This tutorial builds on and expands a number of topics covered in the previous tutorial (including using data and models treated there), and the latter is thus a great way to get up to speed before studying the former.

0.1 Software installation

Minimally, you will need to install the latest version of **R** (available here), the `parallel` package, and the `statnet` packages `relevent`, `sna`, and `network` to run the code presented here (`sna` will automatically install `network` when it is installed).

The full set of installation instructions with details can be found on the `statnet` workshop wiki.

If you have not already downloaded the `statnet` packages, the quickest way to install these (and the other most commonly used packages from the `statnet` suite), is to open an R session and type:

```
install.packages(c("relevent","sna","parallel"))
```

This will also ensure that you have the `parallel` package installed, which may be helpful for speeding up some demonstrations (depending on your hardware and software configuration).

```
library(relevent)
```

```
Loading required package: trust

Loading required package: sna

Loading required package: statnet.common

Attaching package: 'statnet.common'

The following objects are masked from 'package:base':

    attr, order

Loading required package: network

'network' 1.18.1 (2023-01-24), part of the Statnet Project
* 'news(package="network")' for changes since last version
* 'citation("network")' for citation information
* 'https://statnet.org' for help, support, and other information

sna: Tools for Social Network Analysis
Version 2.7-1 created on 2023-01-24.
copyright (c) 2005, Carter T. Butts, University of California-Irvine
 For citation information, type citation("sna").
 Type help(package="sna") to get started.

Loading required package: coda

relevent: Relational Event Models
Version 1.2-1 created on 2023-01-24.
copyright (c) 2007, Carter T. Butts, University of California-
Irvine
 For citation information, type citation("relevent").
 Type help(package="relevent") to get started.
```

```
library(parallel)
```

You can check the version number with:

```
packageVersion("relevent")
```

```
[1] '1.2.1'
```

Throughout, we will set a random seed via `set.seed()` for commands in the tutorial that require simulating random values—this is not necessary, but it (probably) ensures that you will get the same results as the tutorial. To follow the analyses, you will also need to load the workshop data. Assuming that it is in your current path, you can load it like so:

```
load("in_silico_experiments_tutorial.RData")
```

Note that if you put the file in a different location, you may need to alter the above to provide the path to the data file.

Finally, we need to set the number of cores to use for parallel simulation of experiments. Depending on your hardware and operating system, you may be stuck at "1" here (no parallelization), but if you have a compatible OS and multiple cores, it will speed things up quite a lot to increase this. (See `?parallel` for details.) However, if you leave this at 1, no harm will be done.

```
mc.cores<-1          #Set to 1 by default, but consider increasing!
```

Section 1. A Quorum of Concepts Involving *In Silico* Experiments

The basic idea behind *in silico* (i.e., computational) experiments predates modern computing hardware: given a model for a phenomenon of interest, vary the parameters of or inputs to study how the behavior of the model changes. Why would we care about the behavior of a *model*? There are many reasons. Among them: sometimes, the model embodies mechanisms of fundamental interest (that we wish to study); sometimes, the model is being used to drive a decision (or for a similar use), and we want to see how robust it is to changes in parameters or inputs; and sometimes, we think that the model is a sufficiently credible proxy for a real process that its behavior may give us insight into how that process works.

Here, we will focus on a specific class of such experiments: ones in which we begin with an *empirically calibrated model,* and *perturb or modify specific aspects of that model* in order to examine changes in model behavior with respect to a *target outcome.* Experiments of this kind are used throughout the sciences, for many different purposes, but we will be concerned with how to use them to probe social dynamics, in the context of relational event models as implemented in the `statnet` toolchain. We do not thus attempt to give a full (nor rigorous) treatment of the topic here, but limit ourselves to a few simple background ideas that we will use in what follows.

In general, we can think of our experiments as follows. Let $M(\theta, X)$ be a stochastic process (corresponding to a model of interest), that takes a parameter vector θ and covariate set X, and returns a (random) realization. While M could take any form, we are usually interested in models for which the parameters refer to the action of particular predictors or social mechanisms, that are proposed to drive a phenomenon of interest. The realizations generated by M could in principle be a vector of outputs representing individual characteristics, a network, or something else - but here, those outputs will be *event histories* representing timed sequences of interactions among individuals in a social system. Let f represent some *statistic* defined on the outputs - i.e. a property or vector of properties in which we are interested. Our experiments are then concerned with the behavior of $f(M(\theta, X))$, as θ and/or X are varied. In particular, we will begin with some specified model $M(\hat{\theta}_{obs}, X_{obs})$ that has been calibrated to a data set, and then consider how f changes when we deviate from our estimated parameters (or observed covariates). This is accomplished by Monte Carlo methods: that is, we take multiple draws $y_1, \ldots, y_n \sim f(M(\theta, X))$, and compare samples obtained with different values of θ and X. In principle, this is extremely simple, though doing so in a way that yields useful insights (or that can be computed in a reasonable period of time) is not always trivial.

While we again stress that there are diverse approaches to this topic, we can for the most part think of the typical experiment we wish to perform in terms of a basic "recipe:"

1. Given observations Z, select and calibrate model $M(\hat{\theta}_{obs}, X_{obs})$.
2. Choose statistic f of substantive interest.
3. Simulate $f(M(\hat{\theta}_{obs}, X_{obs}))$ and compare to $f(Z)$ to verify *model adequacy* for the target statistic.
4. Identify a set of *conditions*, $(\theta^{(1)}, X^{(1)}), \ldots, (\theta^{(m)}, X^{(m)})$, to be examined.
5. For each condition, draw a sample $y_1^{(i)}, \ldots, y_n^{(i)} \sim f(M(\theta^{(i)}, X^{(i)}))$.
6. Analyze cross-sample variation to understand the impact of θ and X on the outcome of interest under M.

Even this recipe has countless variations! Some - particularly ways of choosing alternative parameters and covariates - are so broadly useful, however, that they are worth noting (and naming). Our nomenclature for these types of designs comes primarily from analogy to experimental methods in the biomedical sciences, where analogous (but *in vivo* and *in vitro*) procedures have been extensively developed to make progress in understanding the behavior of complex biological systems:

- *Knock-out experiments* are experiments in which our conditions involve *removing or deactivating* some mechanism within M, usually by setting some $\theta_i = 0$. By "turning off" select aspects of model behavior while leaving the other parts intact, we can probe the roles of the respective mechanisms in shaping the target outcome.
- *Knock up/down experiments* are experiments in which our conditions involve *strengthening or weakening* the action of some mechanism within M, generally in our context by setting some $\theta_i = \gamma\hat{\theta}_{obs,i}$. By "tuning up" or "tuning down" a specific mechanism, we can obtain a sense not only of that the mechanism may be doing in shaping the target outcome, but what it "could" do (in a hypothetical sense).
- *Knock-in experiments* are experiments in which our conditions involve *adding* a new mechanism to M, generally in our case by taking some θ_i that was implicitly equal to 0 in the calibrated model and setting it to a non-zero value. By introducing a new mechanism to the calibrated set, we can explore how hypothetically interesting (or plausible) processes might interact with those already believed to be present.
- *Context-shifting experiments* are experiments in which our conditions involve changing the covariates (X) that determine the context or "substrate" on which the system acts. Typically, this involves changing covariates from X_{obs} to substantively relevant alternative values. These experiments are often useful for understanding how an empirically calibrated set of mechanisms might play out in a new setting, or probing the potential effects of interventions in which covariates might be deliberately altered.

We will see examples of all of these types of experiments here, and show how they may be implemented for relational event models using `relevent`.

Finally, we note that in this brief discussion we have made no specific reference to how our model was calibrated, nor to the presumptive epistemic properties of the model in question. Much can be said regarding these and other issues, but here we concern ourselves narrowly with the question of how to carry out simple experiments using the `relevent` package. Relatedly, the restrictions of a tutorial format lead us to take a more casual (and less rigorous) approach that would be ideal in an applied setting. These demonstrations, however, introduce the basic tools required to carry out simple *in silico* experiments, which may be elaborated as needed for the occasion.

Section 2. A Quick REM Refresher

Relational events (Butts 2008) are discrete representations of actions, behaviors, state changes, or other phenomena that can be approximated as (1) relational (i.e., connecting one or more *senders* to one or more *receivers*) and (2) well-localized in time, relative to the system of interest. Typically, relational events can be represented as 4-tuples (s, r, c, t), where s is a sender (or set thereof), r is a receiver (or set thereof), c is a vector of event properties, and t is a time point. Often, we study "unvalued," dyadic relational events that can be more simply represented in terms of a single sender, a single receiver, and the time when the event occurs - these will be our focus in this tutorial.

The critical feature of relational events that distinguishes them from ties of the sort studied in social network analysis is that they are - at least approximately - *instantaneous.* Social ties are *temporally extensive:* even in dynamic networks, there is a non-vanishing period of time in which any given tie is present, which allows ties to be *coterminous.* The vast majority of social network modeling and analysis is focused on the structure of such coterminous relationships, and their evolution over time. By contrast, relational events are not coterminous, and do not give rise to instantaneous network structure. They are, however, well-ordered in time. This basic fact leads us to ask very different questions of relational events than are asked of social ties, and it also leads us to very different kinds of models.

Relational events, aptly enough, are studied using *relational event models* (REMs). REMs model relational dynamics in terms of *events* in *continuous time;* if you are familiar with survival, event history, or hazard analysis, the core ideas behind REMs will be natural to you (though different fields tend to use different language to discuss the same concepts, and impose slightly different assumptions, so you may see some things that are not as you expect). The most common approach can be very tersely summarized as follows. At any given time, t, the set of all so-far realized events forms the *event history,* A_t. The set of events that are possible given that event history is then given by $\mathbb{A}(A_t)$ - this lets us accounts for the fact that, in some systems, some events may enable or foreclose the possibility of others. Letting $s(a)$, $r(a)$, and $c(a)$ be the event sender, receiver, and category (if applicable) for event a, and X_a a set of covariates for a, we then write the *hazard* (or *rate*) for a as

$$\lambda_{aA_t\theta} = \begin{cases} \exp\left(\theta^T u\left(s(a), r(a), c(a), X_a, A_t\right)\right) & \text{if } a \in \mathbb{A}(A_t) \\ 0 & \text{otherwise} \end{cases},$$

where $\lambda_{aA_t\theta}$ is the hazard for a given the event history up to the current moment, θ is a vector of parameters, and u is a vector of real-valued statistics that depend on the history and event properties. Intuitively, every unit change in u_i for event a multiplies the hazard of a by $\exp(\theta)$ (which might be a decrease, if $\theta < 0$). Typically, we assume that the hazards remain constant until an event occurs, at which point they may change (based on the above equation). Thus, realized events drive the hazard of the next event, whose realization in turn updates the hazards. (It is possible to relax this assumption, e.g. using exogenous "clock events" (Marcum and Butts 2015) that update the hazards without generating an endogenous event. But here we stick to the basic case.)

Given an observed event history, we may specify a REM by choosing the u functions, and any associated covariates; these reflect the social, cognitive, or other mechanisms proposed to drive the social dynamics. Given such a specification, inference for θ can be performed using maximum likelihood or Bayesian methods. Model selection may likewise be performed using standard statistical methods (e.g., penalized model selection statistics such as the BIC or AICc). In cases where the event timing is not observed, it is still possible to estimate a REM of the form described above (up to a pacing constant) using a marginalized likelihood, so long as the *order* of events is known - this allows REMs to be used with data derived from transcripts or other sources in which event times are not recorded. We direct the interested reader to the associated references for statistical details.

REMs were introduced in Butts (2008); many extensions and other subsequent developments have been made by a growing community of researchers. For those seeking more information on these topics, an accessible overview with references to past and recent work can be found in Butts et al. (2023).

Within `statnet`, inference and simulation for REMs is performed using the `relevent` package. `rem.dyad` is the primary workhorse tool for handling dyadic relational event data, and will be our focus here. In particular, `rem.dyad` has a `simulate` method that allows simulation of event histories from either fitted or *de novo* REMs. Here, we will examine how this capability can be used to perform *in silico* experiments to probe the role of specific mechanisms on social dynamics, or to glean insights that could help inform intervention studies. To this end, we will use models fit to some sample data sets - we generate these below.

2.1 The WTC Police Radio Communication Model

The data set we will use for one of our running examples in this tutorial comes from the World Trade Center radio communication data set coded by Butts, Petrescu-Prahova, and Cross (2007). It consists of radio calls among 37 named communicants belonging to a police unit at the World Trade Center complex on the morning of 9/11/2001. The data is stored in an edgelist (or event list), contained in an object called `WTCPoliceCalls`; printing it should yield output like the following:

```r
head(WTCPoliceCalls)
```

```
  number source recipient
1      1     16        32
2      2     32        16
3      3     16        32
4      4     16        32
5      5     11        32
6      6     11        32
```

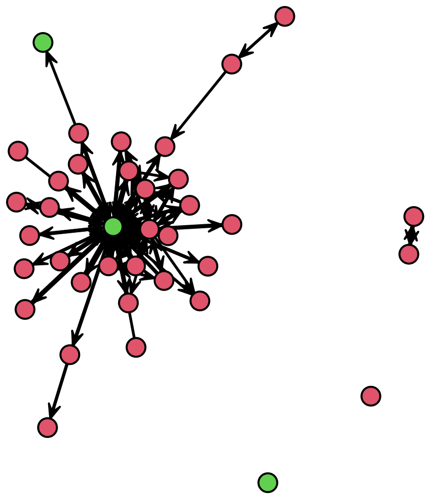
```r
tail(WTCPoliceCalls)
```

```
    number source recipient
476    476     28        16
477    477     16        28
478    478     28        16
479    479     15        16
480    480     32        16
481    481     16        32
```

Note the form of the data: a matrix with the timing information, source (numbered from 1 to 37), and recipient (again numbered from 1 to 37) for each event (i.e., radio call). It is important to note that the WTC radio data was coded from transcripts that lacked detailed timing information; we do not therefore know precisely when these calls were made. We do, however, know the order in which calls were made, and can use this to fit relational event models with `rem.dyad`.

Before analyzing the data, it is helpful to consider what it looks like in time aggregated form. The helper function `as.sociomatrix.eventlist` is useful for this purpose: it converts an event list into a valued sociomatrix, of the form used by other `statnet` routines. Let's convert the data to sociomatrix form, and visualize it using the `gplot` function of the `sna` package:

```r
WTCPoliceNet <- as.sociomatrix.eventlist(WTCPoliceCalls, 37)
gplot(WTCPoliceNet, edge.lwd = WTCPoliceNet^0.75, vertex.col = 2 +
   WTCPoliceIsICR, vertex.cex = 1.25)
```



In this visualization, we have scaled edge widths by communication volume – clearly, some pairs interact much more than others. Note also that we have colored vertices based on whether or not they occupy an institutionalized coordinative role (ICR), as indicated by the vector `WTCPoliceIsICR`. Those for whom this vector is TRUE (green) occupy roles within the police organization that would be expected to participate in coordinative activities; other actors were not identified as occupying such roles, based on the transcript data. We will employ this covariate (as well as various endogenous mechanisms) to model the dynamics of radio communication within the WTC police network.

2.1.1 Fitting a Model to the WTC Data

For a detailed treatment of REM fitting and evaluation for this data using `relevent`, see the `statnet` relational event tutorial. Here, we skip to the chase, noting only a few details that we will want later. In the original study of the WTC networks, Butts (2008) (see also Renshaw et al. (2023)) identifies several plausible drivers of communication among agents in the WTC complex, including

- Pre-existing, institutionalized coordinative roles (ICR), which position some individuals to become central as both targets for and initiators of communicative acts;
- Conversational norms (embodied in radio standard operating procedures (SOP)) that strongly favor turn-taking, “baton passing,” and sequential address;
- Mnemonic effects that make egocentrically recent incoming and outgoing conversation partners favored targets for communication; and
- A “preferential attachment” effect, whereby those involved in a greater share of radio traffic become increasingly salient to others, and thus more attractive as subsequent targets.

A model based on these effects can be fit to the WTC police call data as follows:

```
set.seed(13)  #To allow later results to be reproduced...
wtcfit <- rem.dyad(WTCPoliceCalls, n = 37, effects = c("CovInt",
   "PSAB-BA", "PSAB-BY", "PSAB-AY", "RRecSnd", "RSndSnd", "NTDegRec"),
   covar = list(CovInt = WTCPoliceIsICR), hessian = TRUE)
```

```
Prepping edgelist.
Checking/prepping covariates.
Computing preliminary statistics
Fitting model
Obtaining goodness-of-fit statistics
```

```
summary(wtcfit)  #Show the parameter estimates, and related quantities
```

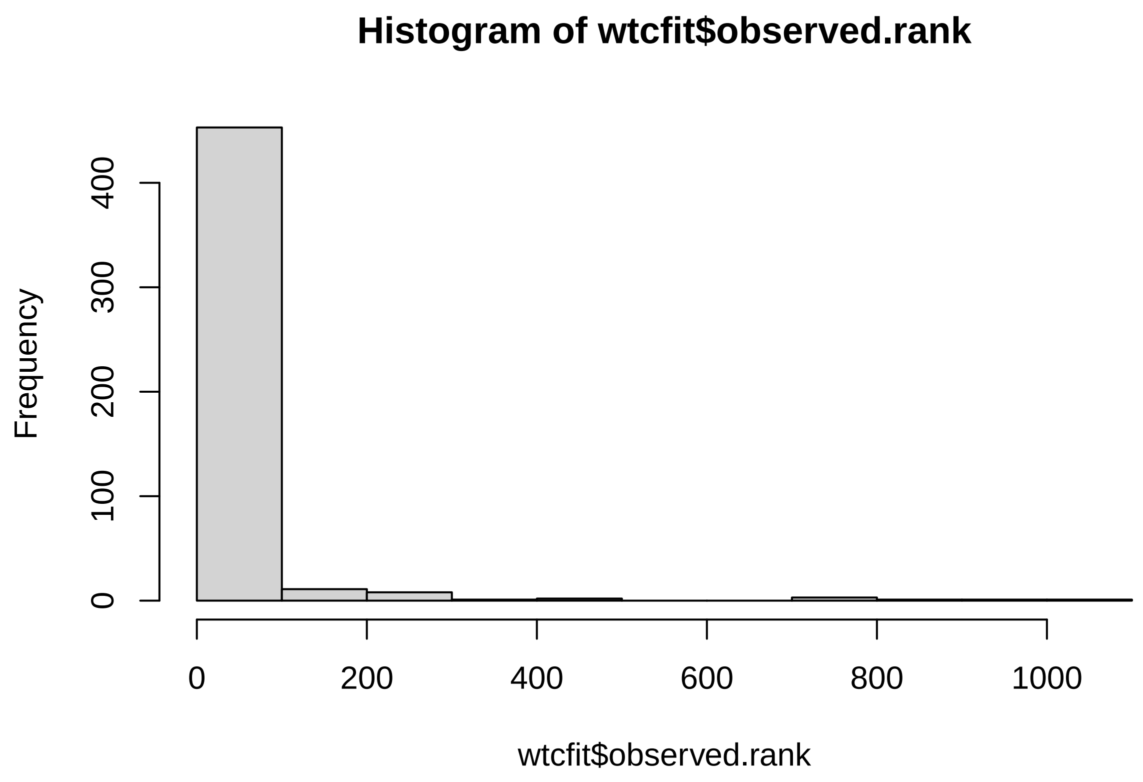
```
Relational Event Model (Ordinal Likelihood)

         Estimate Std.Err Z value  Pr(>|z|)
NTDegRec  3.13454 0.56678  5.5305 3.194e-08 ***
RRecSnd   2.02903 0.28500  7.1194 1.084e-12 ***
RSndSnd   0.87115 0.23846  3.6533 0.0002589 ***
CovInt.1  0.70734 0.16400  4.3129 1.611e-05 ***
PSAB-BA   5.32576 0.18236 29.2042 < 2.2e-16 ***
PSAB-BY   1.86023 0.26322  7.0674 1.579e-12 ***
PSAB-AY   1.64806 0.31092  5.3005 1.155e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Null deviance: 6921.048 on 481 degrees of freedom
Residual deviance: 2276.263 on 474 degrees of freedom
    Chi-square: 4644.785 on 7 degrees of freedom, asymptotic p-value 0
AIC: 2290.263 AICC: 2290.5 BIC: 2319.494
```

The `rem.dyad` command fits the model, using the event history in `WTCPoliceCalls`; we must tell it how many nodes there are (`n`), and what effects (u functions) to put in the model (`effects`). The `covar` argument allows us to pass covariate information, here the vector of ICR indicators `WTCPoliceIsICR`. (See `?rem.dyad` or the above-mentioned tutorial for additional details.) The model output gives us the θ estimates for each effect (here, posterior modes under very diffuse 0-centered priors), and associated estimates of uncertainty (technically, estimated posterior standard deviations). In this case, the estimated parameters are all quite far from 0, relative to their uncertainties (as shown by the Z values, and associated frequentist tests). The effects may be interpreted very broadly as follows:

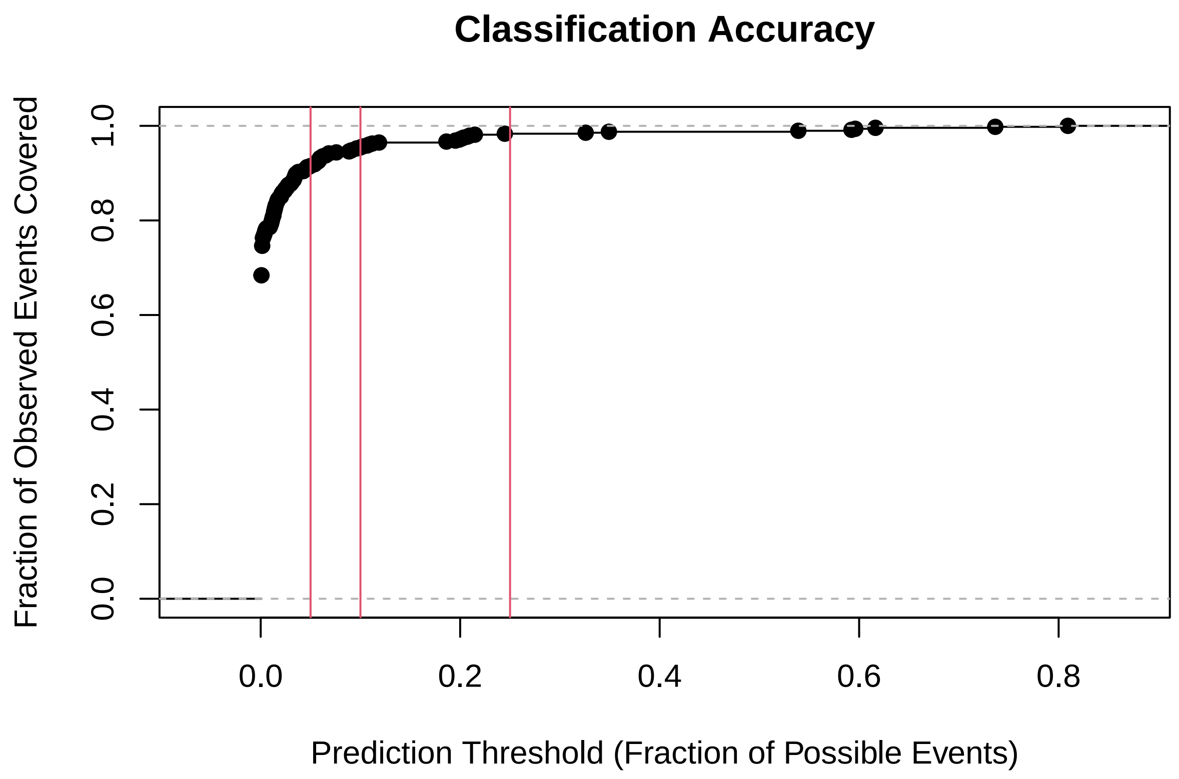
- There is indeed a preferential attachment effect (`NTDegRec`), with those having a higher fraction of total traffic at any given time being more favored communication targets;
- Ego's more recent incoming (`RRecSnd`) and outgoing (`RSndSnd`) communication partners are targeted by ego at higher rates - but this effect is much greater for incoming than outgoing partners;
- There is a modest tendency for ICRs to interact and be interacted with at higher rates;
- Turn-taking reciprocity (`PSAB-BA`) has a profound effect on communication dynamics, with baton passing (`PSAB-BY`) and sequential address (`PSAB-AY`) being weaker but prevalent.

2.1.2 Assessing the WTC Model The `statnet` relational event tutorial discusses model assessment (including the assessment of this model) at some length; although this is an important aspect of model building, here we simply examine a few basic indicators. One useful tool for adequacy assessment is to consider the rank of the observed events in the predicted rate structure: that is, we ask to what extent the events viewed most likely to occur are in fact those that are observed.

```
hist(wtcfit$observed.rank)  #Histogram of ranks (low is good)
```



```
# Sometimes useful to plot the ECDF of the observed
# ranks....
plot(ecdf(wtcfit$observed.rank/(37 * 36)), xlab = "Prediction Threshold (Fraction of Possible Events)",
   ylab = "Fraction of Observed Events Covered", main = "Classification Accuracy")
abline(v = c(0.05, 0.1, 0.25), col = 2)
```



The model does quite well. As the above indicates, we sometimes (in fact often) manage to get things exactly right: that is, the event predicted most likely to be the next in the sequence is in fact the one that is observed. Examining the match rate is a very strict notion of adequacy, but can be useful for assessing models that are strongly predictive.

```r
wtcfit$predicted.match  #Exactly correct src/target
```

```
      source recipient
 [1,]  FALSE     FALSE
 [2,]   TRUE      TRUE
 [3,]   TRUE      TRUE
 [4,]  FALSE     FALSE
 [5,]  FALSE     FALSE
 [6,]  FALSE     FALSE
 [7,]  FALSE     FALSE
 [8,]  FALSE     FALSE
 [9,]  FALSE     FALSE
[10,]  FALSE     FALSE
[11,]   TRUE      TRUE
[12,]  FALSE      TRUE
[13,]  FALSE     FALSE
[14,]  FALSE     FALSE
[15,]   TRUE      TRUE
[16,]   TRUE      TRUE
[17,]   TRUE      TRUE
[18,]   TRUE      TRUE
[19,]   TRUE      TRUE
```

```
[20,]  FALSE       TRUE
[21,]  FALSE      FALSE
[22,]  FALSE      FALSE
[23,]  FALSE      FALSE
[24,]   TRUE       TRUE
[25,]   TRUE       TRUE
[26,]   TRUE       TRUE
[27,]  FALSE      FALSE
[28,]   TRUE       TRUE
[29,]   TRUE       TRUE
[30,]  FALSE      FALSE
[31,]   TRUE       TRUE
[32,]   TRUE      FALSE
[33,]  FALSE       TRUE
[34,]   TRUE       TRUE
[35,]   TRUE       TRUE
[36,]   TRUE       TRUE
[37,]  FALSE      FALSE
[38,]   TRUE       TRUE
[39,]   TRUE       TRUE
[40,]  FALSE       TRUE
[41,]  FALSE      FALSE
[42,]   TRUE       TRUE
[43,]   TRUE       TRUE
[44,]  FALSE       TRUE
[45,]  FALSE      FALSE
[46,]   TRUE       TRUE
[47,]   TRUE       TRUE
[48,]   TRUE       TRUE
[49,]   TRUE       TRUE
[50,]   TRUE       TRUE
[51,]  FALSE      FALSE
[52,]  FALSE      FALSE
[53,]  FALSE      FALSE
[54,]   TRUE       TRUE
[55,]   TRUE       TRUE
[56,]   TRUE       TRUE
[57,]   TRUE       TRUE
[58,]   TRUE       TRUE
[59,]  FALSE       TRUE
[60,]   TRUE       TRUE
[61,]  FALSE       TRUE
[62,]   TRUE       TRUE
[63,]   TRUE       TRUE
[64,]   TRUE       TRUE
[65,]  FALSE       TRUE
[66,]   TRUE       TRUE
[67,]   TRUE       TRUE
[68,]  FALSE      FALSE
[69,]   TRUE       TRUE
[70,]   TRUE       TRUE
[71,]   TRUE       TRUE
[72,]   TRUE       TRUE
[73,]  FALSE      FALSE
```

```
 [74,]   TRUE       TRUE
 [75,]   TRUE       TRUE
 [76,]   TRUE       TRUE
 [77,]  FALSE      FALSE
 [78,]   TRUE       TRUE
 [79,]   TRUE       TRUE
 [80,]   TRUE       TRUE
 [81,]   TRUE       TRUE
 [82,]   TRUE       TRUE
 [83,]  FALSE      FALSE
 [84,]   TRUE      FALSE
 [85,]   TRUE       TRUE
 [86,]   TRUE       TRUE
 [87,]   TRUE       TRUE
 [88,]   TRUE       TRUE
 [89,]  FALSE      FALSE
 [90,]   TRUE       TRUE
 [91,]   TRUE       TRUE
 [92,]   TRUE       TRUE
 [93,]   TRUE       TRUE
 [94,]  FALSE       TRUE
 [95,]   TRUE       TRUE
 [96,]   TRUE       TRUE
 [97,]   TRUE      FALSE
 [98,]   TRUE       TRUE
 [99,]  FALSE      FALSE
[100,]   TRUE       TRUE
[101,]   TRUE       TRUE
[102,]   TRUE       TRUE
[103,]   TRUE       TRUE
[104,]   TRUE       TRUE
[105,]  FALSE       TRUE
[106,]  FALSE      FALSE
[107,]   TRUE       TRUE
[108,]   TRUE       TRUE
[109,]   TRUE       TRUE
[110,]  FALSE       TRUE
[111,]   TRUE       TRUE
[112,]   TRUE       TRUE
[113,]   TRUE       TRUE
[114,]  FALSE      FALSE
[115,]   TRUE       TRUE
[116,]   TRUE       TRUE
[117,]   TRUE       TRUE
[118,]  FALSE      FALSE
[119,]  FALSE      FALSE
[120,]   TRUE       TRUE
[121,]   TRUE       TRUE
[122,]   TRUE       TRUE
[123,]  FALSE       TRUE
[124,]   TRUE       TRUE
[125,]  FALSE      FALSE
[126,]  FALSE      FALSE
[127,]  FALSE      FALSE
```

```
[128,]   TRUE     FALSE
[129,]  FALSE      TRUE
[130,]   TRUE      TRUE
[131,]   TRUE      TRUE
[132,]   TRUE      TRUE
[133,]   TRUE      TRUE
[134,]  FALSE      TRUE
[135,]   TRUE      TRUE
[136,]   TRUE      TRUE
[137,]   TRUE      TRUE
[138,]  FALSE     FALSE
[139,]  FALSE     FALSE
[140,]  FALSE     FALSE
[141,]  FALSE     FALSE
[142,]  FALSE     FALSE
[143,]   TRUE      TRUE
[144,]   TRUE      TRUE
[145,]   TRUE      TRUE
[146,]  FALSE     FALSE
[147,]   TRUE      TRUE
[148,]   TRUE      TRUE
[149,]   TRUE      TRUE
[150,]  FALSE      TRUE
[151,]   TRUE      TRUE
[152,]   TRUE      TRUE
[153,]   TRUE     FALSE
[154,]  FALSE      TRUE
[155,]   TRUE      TRUE
[156,]   TRUE      TRUE
[157,]   TRUE      TRUE
[158,]   TRUE      TRUE
[159,]   TRUE      TRUE
[160,]   TRUE      TRUE
[161,]   TRUE      TRUE
[162,]  FALSE      TRUE
[163,]   TRUE      TRUE
[164,]   TRUE      TRUE
[165,]   TRUE      TRUE
[166,]   TRUE      TRUE
[167,]  FALSE     FALSE
[168,]  FALSE     FALSE
[169,]   TRUE     FALSE
[170,]   TRUE      TRUE
[171,]   TRUE      TRUE
[172,]  FALSE      TRUE
[173,]   TRUE      TRUE
[174,]   TRUE      TRUE
[175,]   TRUE      TRUE
[176,]  FALSE      TRUE
[177,]   TRUE      TRUE
[178,]   TRUE      TRUE
[179,]   TRUE      TRUE
[180,]   TRUE      TRUE
[181,]   TRUE      TRUE
```

```
[182,]   TRUE       TRUE
[183,]  FALSE      FALSE
[184,]   TRUE       TRUE
[185,]   TRUE      FALSE
[186,]  FALSE      FALSE
[187,]   TRUE       TRUE
[188,]   TRUE       TRUE
[189,]  FALSE      FALSE
[190,]   TRUE       TRUE
[191,]  FALSE      FALSE
[192,]   TRUE       TRUE
[193,]   TRUE       TRUE
[194,]   TRUE       TRUE
[195,]   TRUE       TRUE
[196,]   TRUE       TRUE
[197,]   TRUE       TRUE
[198,]   TRUE       TRUE
[199,]   TRUE       TRUE
[200,]   TRUE       TRUE
[201,]   TRUE       TRUE
[202,]  FALSE       TRUE
[203,]   TRUE       TRUE
[204,]  FALSE       TRUE
[205,]   TRUE       TRUE
[206,]   TRUE       TRUE
[207,]   TRUE       TRUE
[208,]  FALSE       TRUE
[209,]   TRUE       TRUE
[210,]   TRUE       TRUE
[211,]   TRUE      FALSE
[212,]   TRUE       TRUE
[213,]  FALSE      FALSE
[214,]  FALSE       TRUE
[215,]   TRUE       TRUE
[216,]   TRUE       TRUE
[217,]   TRUE       TRUE
[218,]   TRUE       TRUE
[219,]   TRUE       TRUE
[220,]  FALSE      FALSE
[221,]   TRUE       TRUE
[222,]   TRUE       TRUE
[223,]   TRUE       TRUE
[224,]   TRUE       TRUE
[225,]  FALSE      FALSE
[226,]   TRUE       TRUE
[227,]   TRUE       TRUE
[228,]   TRUE       TRUE
[229,]   TRUE      FALSE
[230,]   TRUE       TRUE
[231,]   TRUE       TRUE
[232,]   TRUE       TRUE
[233,]   TRUE       TRUE
[234,]   TRUE       TRUE
[235,]   TRUE       TRUE
```

```
[236,]   TRUE      TRUE
[237,]   TRUE      TRUE
[238,]   TRUE      TRUE
[239,]   TRUE     FALSE
[240,]   TRUE      TRUE
[241,]   TRUE      TRUE
[242,]   TRUE      TRUE
[243,]   TRUE     FALSE
[244,]  FALSE     FALSE
[245,]  FALSE     FALSE
[246,]  FALSE      TRUE
[247,]   TRUE      TRUE
[248,]   TRUE      TRUE
[249,]   TRUE      TRUE
[250,]  FALSE     FALSE
[251,]   TRUE      TRUE
[252,]   TRUE      TRUE
[253,]   TRUE      TRUE
[254,]   TRUE      TRUE
[255,]   TRUE      TRUE
[256,]  FALSE     FALSE
[257,]   TRUE      TRUE
[258,]   TRUE      TRUE
[259,]   TRUE      TRUE
[260,]  FALSE     FALSE
[261,]   TRUE      TRUE
[262,]   TRUE      TRUE
[263,]   TRUE      TRUE
[264,]   TRUE      TRUE
[265,]   TRUE      TRUE
[266,]   TRUE      TRUE
[267,]   TRUE      TRUE
[268,]   TRUE      TRUE
[269,]  FALSE     FALSE
[270,]   TRUE      TRUE
[271,]   TRUE      TRUE
[272,]   TRUE      TRUE
[273,]   TRUE      TRUE
[274,]   TRUE      TRUE
[275,]   TRUE      TRUE
[276,]  FALSE     FALSE
[277,]   TRUE      TRUE
[278,]   TRUE      TRUE
[279,]   TRUE      TRUE
[280,]   TRUE      TRUE
[281,]   TRUE      TRUE
[282,]   TRUE      TRUE
[283,]   TRUE      TRUE
[284,]   TRUE      TRUE
[285,]   TRUE      TRUE
[286,]   TRUE      TRUE
[287,]   TRUE      TRUE
[288,]   TRUE      TRUE
[289,]  FALSE     FALSE
```

```
[290,]   TRUE     FALSE
[291,]   TRUE      TRUE
[292,]   TRUE      TRUE
[293,]   TRUE      TRUE
[294,]   TRUE      TRUE
[295,]   TRUE      TRUE
[296,]   TRUE      TRUE
[297,]   TRUE      TRUE
[298,]   TRUE      TRUE
[299,]   TRUE      TRUE
[300,]   TRUE      TRUE
[301,]   TRUE      TRUE
[302,]   TRUE      TRUE
[303,]  FALSE      TRUE
[304,]   TRUE      TRUE
[305,]   TRUE      TRUE
[306,]  FALSE     FALSE
[307,]   TRUE      TRUE
[308,]   TRUE      TRUE
[309,]  FALSE     FALSE
[310,]   TRUE      TRUE
[311,]   TRUE      TRUE
[312,]   TRUE     FALSE
[313,]  FALSE      TRUE
[314,]   TRUE      TRUE
[315,]   TRUE      TRUE
[316,]   TRUE      TRUE
[317,]  FALSE      TRUE
[318,]   TRUE      TRUE
[319,]   TRUE      TRUE
[320,]   TRUE      TRUE
[321,]   TRUE      TRUE
[322,]   TRUE      TRUE
[323,]   TRUE      TRUE
[324,]  FALSE     FALSE
[325,]   TRUE      TRUE
[326,]  FALSE     FALSE
[327,]   TRUE      TRUE
[328,]  FALSE     FALSE
[329,]   TRUE      TRUE
[330,]   TRUE      TRUE
[331,]   TRUE      TRUE
[332,]  FALSE      TRUE
[333,]   TRUE      TRUE
[334,]  FALSE     FALSE
[335,]   TRUE      TRUE
[336,]   TRUE      TRUE
[337,]   TRUE      TRUE
[338,]   TRUE      TRUE
[339,]   TRUE      TRUE
[340,]  FALSE     FALSE
[341,]   TRUE      TRUE
[342,]   TRUE      TRUE
[343,]   TRUE      TRUE
```

```
[344,]  FALSE     FALSE
[345,]   TRUE      TRUE
[346,]   TRUE      TRUE
[347,]  FALSE     FALSE
[348,]  FALSE     FALSE
[349,]  FALSE     FALSE
[350,]   TRUE      TRUE
[351,]   TRUE      TRUE
[352,]   TRUE      TRUE
[353,]  FALSE      TRUE
[354,]   TRUE      TRUE
[355,]   TRUE      TRUE
[356,]   TRUE      TRUE
[357,]  FALSE     FALSE
[358,]   TRUE      TRUE
[359,]  FALSE     FALSE
[360,]  FALSE      TRUE
[361,]  FALSE     FALSE
[362,]   TRUE      TRUE
[363,]   TRUE      TRUE
[364,]   TRUE      TRUE
[365,]   TRUE      TRUE
[366,]   TRUE     FALSE
[367,]   TRUE      TRUE
[368,]   TRUE      TRUE
[369,]   TRUE      TRUE
[370,]  FALSE     FALSE
[371,]  FALSE     FALSE
[372,]   TRUE     FALSE
[373,]  FALSE     FALSE
[374,]  FALSE      TRUE
[375,]   TRUE      TRUE
[376,]   TRUE      TRUE
[377,]   TRUE     FALSE
[378,]   TRUE      TRUE
[379,]  FALSE     FALSE
[380,]   TRUE      TRUE
[381,]   TRUE      TRUE
[382,]   TRUE      TRUE
[383,]  FALSE     FALSE
[384,]   TRUE      TRUE
[385,]  FALSE     FALSE
[386,]   TRUE      TRUE
[387,]  FALSE     FALSE
[388,]  FALSE     FALSE
[389,]   TRUE      TRUE
[390,]   TRUE      TRUE
[391,]   TRUE      TRUE
[392,]   TRUE      TRUE
[393,]   TRUE      TRUE
[394,]   TRUE      TRUE
[395,]  FALSE     FALSE
[396,]   TRUE      TRUE
[397,]   TRUE      TRUE
```

```
[398,]   TRUE       TRUE
[399,]  FALSE      FALSE
[400,]   TRUE       TRUE
[401,]  FALSE      FALSE
[402,]   TRUE       TRUE
[403,]   TRUE       TRUE
[404,]   TRUE       TRUE
[405,]   TRUE       TRUE
[406,]  FALSE      FALSE
[407,]   TRUE       TRUE
[408,]   TRUE       TRUE
[409,]   TRUE       TRUE
[410,]  FALSE      FALSE
[411,]   TRUE      FALSE
[412,]   TRUE       TRUE
[413,]   TRUE       TRUE
[414,]   TRUE       TRUE
[415,]   TRUE       TRUE
[416,]  FALSE       TRUE
[417,]   TRUE       TRUE
[418,]   TRUE       TRUE
[419,]  FALSE      FALSE
[420,]  FALSE      FALSE
[421,]   TRUE       TRUE
[422,]   TRUE       TRUE
[423,]   TRUE       TRUE
[424,]   TRUE      FALSE
[425,]   TRUE       TRUE
[426,]   TRUE       TRUE
[427,]   TRUE      FALSE
[428,]   TRUE       TRUE
[429,]   TRUE       TRUE
[430,]   TRUE       TRUE
[431,]   TRUE       TRUE
[432,]   TRUE       TRUE
[433,]   TRUE       TRUE
[434,]  FALSE      FALSE
[435,]   TRUE       TRUE
[436,]   TRUE       TRUE
[437,]   TRUE       TRUE
[438,]   TRUE       TRUE
[439,]  FALSE      FALSE
[440,]   TRUE       TRUE
[441,]   TRUE       TRUE
[442,]   TRUE       TRUE
[443,]  FALSE      FALSE
[444,]   TRUE       TRUE
[445,]   TRUE       TRUE
[446,]   TRUE       TRUE
[447,]   TRUE       TRUE
[448,]   TRUE       TRUE
[449,]  FALSE      FALSE
[450,]   TRUE       TRUE
[451,]   TRUE       TRUE
```

```
[452,]  FALSE      TRUE
[453,]  FALSE     FALSE
[454,]  FALSE     FALSE
[455,]  FALSE     FALSE
[456,]   TRUE      TRUE
[457,]   TRUE      TRUE
[458,]   TRUE      TRUE
[459,]  FALSE     FALSE
[460,]   TRUE      TRUE
[461,]  FALSE     FALSE
[462,]   TRUE      TRUE
[463,]   TRUE      TRUE
[464,]   TRUE      TRUE
[465,]   TRUE      TRUE
[466,]   TRUE      TRUE
[467,]  FALSE     FALSE
[468,]   TRUE      TRUE
[469,]   TRUE      TRUE
[470,]   TRUE      TRUE
[471,]   TRUE      TRUE
[472,]   TRUE      TRUE
[473,]  FALSE      TRUE
[474,]   TRUE      TRUE
[475,]   TRUE      TRUE
[476,]  FALSE     FALSE
[477,]   TRUE      TRUE
[478,]   TRUE      TRUE
[479,]  FALSE     FALSE
[480,]  FALSE     FALSE
[481,]   TRUE      TRUE
```

```
mean(apply(wtcfit$predicted.match, 1, any))  #Fraction for which something is right
```

```
[1] 0.7941788
```

```
mean(apply(wtcfit$predicted.match, 1, all))  #Fraction entirely right
```

```
[1] 0.6839917
```

```
colMeans(wtcfit$predicted.match)  #Fraction src/target, respectively
```

```
   source recipient
0.7234927 0.7546778
```

Despite its simplicity, this and other checks (not shown) suggest that the model seems to do extremely well at recovering the microdynamics of the WTC radio communication networks. We thus use it for our experiments, below.

2.2 The McFarland Classroom Model

As a second base case, we will make use of data collected by Dan McFarland (and published in Bender-deMoll and McFarland (2006)) on interaction among students and instructors within a high school classroom. (Note that the data employed here has been slightly modified from the original for illustrative purposes, in that small timing adjustments have been made to separate closely spaced events; those interested in using it for purposes other than practice are directed to the above paper in the *Journal of Social Structure*.) To see the event data itself, we may print it as follows:

```
head(Class)
```

```
  StartTime FromId ToId
1     0.135     14   12
2     0.270     12   14
3     0.405     18   12
4     0.540     12   18
5     0.675      1   12
6     0.810     12    1
```

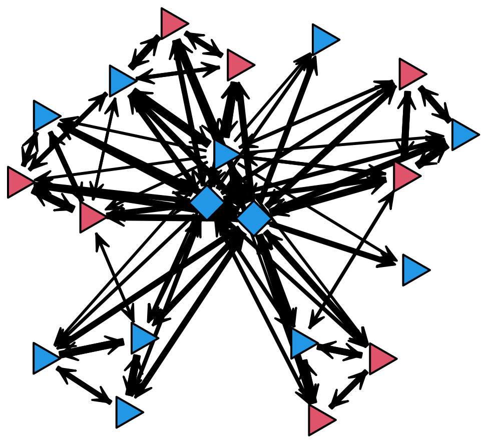
```
tail(Class)
```

```
    StartTime FromId ToId
687    50.426      1    3
688    50.547      3    1
689    50.668      6   17
690    50.789      6   17
691    50.910     17    6
692    50.920     NA   NA
```

As before, we have three columns: the event time, the event source (numbered from 1 to 20), and the event target (again, numbered 1 to 20). In this case, event time is given in increments of minutes from onset of observation. Note that the last row of the event list contains the time at which observation was terminated; it (and only it!) is allowed to contain `NA`s, since it has no meaning except to set the period during which events could have occurred. Where exact timing is used, the final entry in the edgelist is always interpreted in this way, and any source/target information on this row is ignored.

In addition to the `Class` edgelist, we also observe the covariates `ClassIsTeacher` (an indicator for instructor role) and `ClassIsFemale` (an indicator for gender). Visualizing the data in time-aggregate form gives us the following:

```
ClassNet <- as.sociomatrix.eventlist(Class, 20)
gplot(ClassNet, vertex.col = 4 - 2 * ClassIsFemale, vertex.sides = 3 +
   ClassIsTeacher, vertex.cex = 2, edge.lwd = ClassNet^0.75)
```



A dynamic visualization for this data is also available in the above-cited paper, and is well worth examining! (The `ndtv` package in `statnet` can be used to produce visualizations of this kind.)

2.2.1 Fitting a Model to the Classroom Data As above, we work with a model that we might obtain at the end of our modeling process, that incorporates a number of effects. These include the following:

- A basic pacing effect, and both sender and receiver effects for teachers (versus students);
- Egocentric recency effects for incoming and outgoing communication;
- Effects for classroom conversational norms, including turn-taking, baton passing, and sequential address.

(Note that a gender effect was evaluated, but not selected in the final model. However, we will use the covariate in some of our investigations.) We fit the model like so:

```
set.seed(13)  #To ensure that our later results can be reproduced
classfit <- rem.dyad(Class, n = 20, effects = c("CovSnd", "CovRec",
   "RRecSnd", "RSndSnd", "PSAB-BA", "PSAB-AY", "PSAB-BY"), covar = list(CovSnd = cbind(ClassIntercept,
   ClassIsTeacher), CovRec = ClassIsTeacher), ordinal = FALSE,
   hessian = TRUE)
```

```
Prepping edgelist.
Checking/prepping covariates.
Computing preliminary statistics
Fitting model
Obtaining goodness-of-fit statistics
```

```
summary(classfit)
```

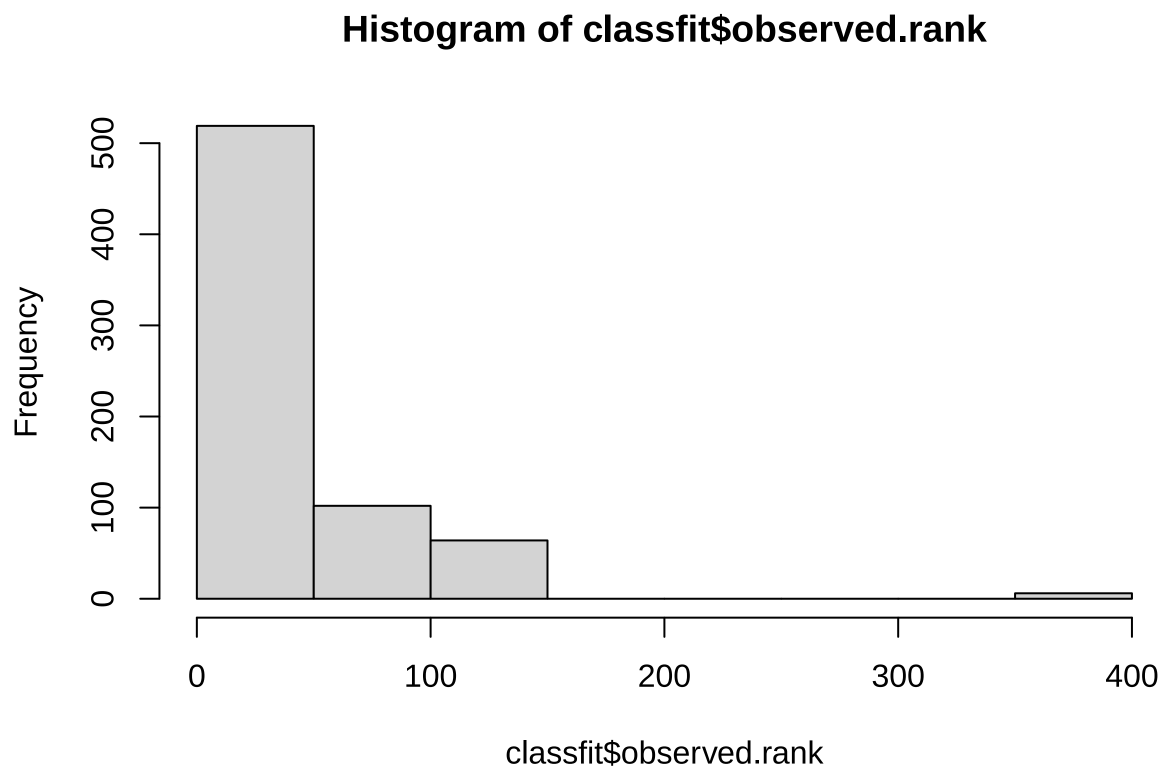
```
Relational Event Model (Temporal Likelihood)
```

```
          Estimate   Std.Err  Z value  Pr(>|z|)
RRecSnd   2.430697  0.155292  15.6524 < 2.2e-16 ***
RSndSnd  -0.984633  0.144654  -6.8068 9.979e-12 ***
CovSnd.1 -4.983918  0.084196 -59.1943 < 2.2e-16 ***
CovSnd.2  1.257292  0.084967  14.7975 < 2.2e-16 ***
CovRec.1 -0.745119  0.136612  -5.4543 4.917e-08 ***
PSAB-BA   4.623684  0.137503  33.6261 < 2.2e-16 ***
PSAB-BY   1.677832  0.164940  10.1724 < 2.2e-16 ***
PSAB-AY   2.870517  0.103103  27.8411 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Null deviance: 5987.221 on 691 degrees of freedom
Residual deviance: 2803.662 on 684 degrees of freedom
    Chi-square: 3183.558 on 7 degrees of freedom, asymptotic p-value 0
AIC: 2819.662 AICC: 2819.874 BIC: 2855.968
```

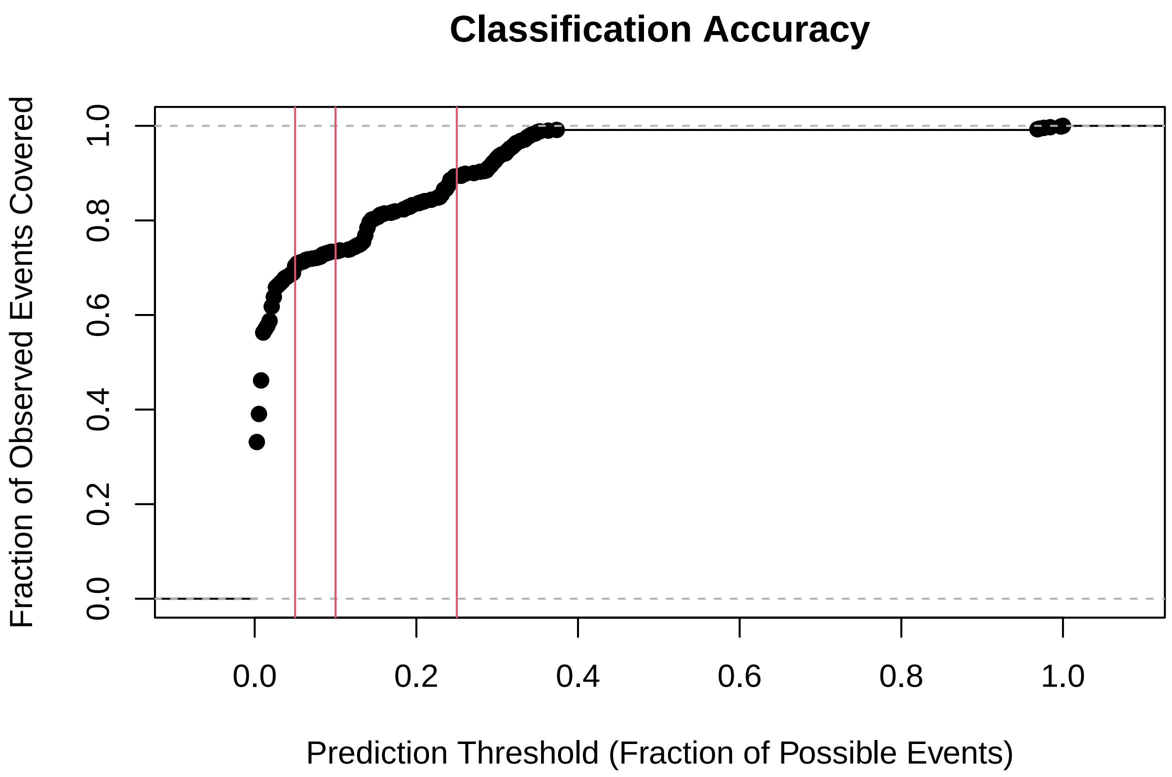
The model effects may be broadly interpreted as follows: * Egocentically recent incoming partners are high-rate targets for outgoing communication, but ego tends to "walk" communication around the room by being less likely to target those to whom they have most recently directed communicative acts; * Teachers initiate communications at a higher rate than students, but are somewhat less attractive targets; * As with the WTC network, turn-taking is important (albeit somewhat less so), with sequential address playing a much larger role in the classroom and baton passing also being common.

2.2.2 Assessing the Classroom Model As above, we only consider a minimal assessment here, but verify that the model is able to capture local dynamics sufficiently to be useful. We again examine the ranks of our predicted events, and the extent to which the events that were observed were seen as likely:

```r
hist(classfit$observed.rank)  #Histogram of ranks (low is good)
```



```r
# Plot the ECDF of the observed ranks
plot(ecdf(classfit$observed.rank/(20 * 19)), xlab = "Prediction Threshold (Fraction of Possible Events)
   ylab = "Fraction of Observed Events Covered", main = "Classification Accuracy")
abline(v = c(0.05, 0.1, 0.25), col = 2)
```



The model does not do quite as well as the WTC model, but we do see that most events are in the top fraction of those predicted. Although the exact match statistic is a difficult one (we would be surprised if such a simple model were always able to tell who will interact with whom next in a high school classroom), we also see that the model is reasonable:

```
mean(apply(classfit$predicted.match, 1, all))   #Exactly right about 33%
```

```
[1] 0.3299566
```

```
mean(apply(classfit$predicted.match, 1, any))   #Get one party exactly right 52%
```

```
[1] 0.5166425
```

```
colMeans(classfit$predicted.match)   #Better at sender than receiver!
```

```
   FromId      ToId
0.5050651 0.3415340
```

Given more data, the model could probably be improved (see e.g. (DuBois et al. 2013)), but it is useful enough for present purposes.

Section 3. *In Silico* Knockout (and Up, and Down) Experiments with REMs

To perform *in silico* experiments, we need to be able to simulate realizations (interaction sequences) from a REM. In addition to fitting REMs, `relevent` has tools for simulating from them. These work a bit like the `simulate` commands in the `ergm` library, in that they can be used in two modes: we can simulate draws from a fitted `rem.dyad` model; or we can simulate draws from an *a priori* specified model. For now, let's consider this first case.

The syntax for the `rem.dyad` `simulate` method is as follows:

```
simulate(object, nsim = object$m, seed = NULL, coef = NULL, covar = NULL,
   verbose = FALSE, ...)
```

`object` here is our fitted model object, `nsim` is the number of events to draw from the model (the length of the event series to simulate), `seed` is an optional random number seed to specify, `coef` is a (here optional) coefficient vector, `covar` is our usual covariate list, and `verbose` says whether we want to print tracking information. By default, the coefficients used are taken from the fitted model, but specifying `coef` will allow them to be overridden (a useful tool for performing scenario analyses, as illustrated below). Likewise, the function will by default simulate as many events as were in the original data, but this can be altered by changing `nsim`. Note that we *do* have to specify any covariates being used when simulating, both because `rem.dyad` does not save the input covariates, and because (even if it did) the size of the covariate set in some cases depends on the number of events to be produced.

Let's begin with a simple example: simulating a synthetic replicate of our original data, using our final model. For this, we only need pass our model, and the covariates used:

```
set.seed(13)
simwtc <- simulate(wtcfit, covar = list(CovInt = WTCPoliceIsICR),
   verbose = TRUE)
```

```
Working on event 25 of 481
Working on event 50 of 481
Working on event 75 of 481
Working on event 100 of 481
Working on event 125 of 481
Working on event 150 of 481
Working on event 175 of 481
Working on event 200 of 481
Working on event 225 of 481
Working on event 250 of 481
Working on event 275 of 481
Working on event 300 of 481
Working on event 325 of 481
Working on event 350 of 481
Working on event 375 of 481
Working on event 400 of 481
Working on event 425 of 481
Working on event 450 of 481
Working on event 475 of 481
```

We now have a simulated event sequence from the `wtcfit` model! Let's see what it looks like:

```
head(simwtc)
```

```
             [,1] [,2] [,3]
[1,] 0.0002470765   16   32
[2,] 0.0003174985   32   16
[3,] 0.0003176928   16   32
[4,] 0.0003184215   32   16
[5,] 0.0003284832   16   32
[6,] 0.0003572302   32   16
```

```
tail(simwtc)
```

```
             [,1] [,2] [,3]
[476,] 0.05079416   35    3
[477,] 0.05082861    3   35
```

```
[478,] 0.05102184   35    3
[479,] 0.05120952    3   35
[480,] 0.05175625   21   11
[481,] 0.05180419   25   24
```

As we can see, we now have an event list that looks just like our original data (but that is synthetic). Such synthetic replicates can be used for many purposes, including exploratory simulation, model adequacy checking, and aiding in model interpretation. Below, we use them to illustrate simple *in silico* "knock-out," "knock-up," and "knock-down" experiments.

3.1 Probing the Effect of Turn-Taking on Coordinator Emergence via Knockout

A distinctive aspect of the WTC radio communication networks (including the one studied here) is the central role of *coordinators* in holding the network together: a small number of individuals are responsible for the majority of communication volume, and they play a pivotal role in coordinating action by brokering between third parties. What drives the emergence of these coordinator roles? This is a complex question, WTC communication dynamics are clearly driven by many, interacting factors. To help make sense of such complexities, it can sometimes be useful to probe the role of specific mechanisms using *in silico knock-out* experiments.

In an *in silico* knock-out experiment, we take one or more specific mechanisms in our model and remove it, *leaving all other mechanisms as they are.* We then simulate realizations from our model with the selected mechanisms "knocked out," comparing them with the full model (our control). The difference in behavior can provide insight into the role of the knocked out mechanisms in the full model, just as differences between the development of mice mouse with a knocked-out gene versus control mice provides insight into the role of the gene in development.

It is important to emphasize that we do not refit the model with the terms removed, just as we do not use evolutionary selection to breed mice that are adapted to their knocked-out gene in a biomedical knock-out experiment: that would probe whether the remaining effects are *sufficient* to obtain the targeted behavior, but would not probe the role of the targeted effects on that behavior *in the data-selected model* (our current best proxy for how the social dynamics operate). However, models of that kind can sometimes be useful in and of themselves as a point of comparison. For instance, in the WTC case, a natural hypothesis might be that coordinators are simply ICRs, and that emergent coordination arises directly from differential ICR activity. To probe this, may fit a "sufficiency" model containing only this effect, again to serve as a basis for comparison. This is not a knock-out model, because we are not leaving the covariate effect as it was in the joint model - it instead represents the best model that can be had given the single effect. (In the biomedical analogy, this would be like knocking out all genes for some pathway other than one, and then breeding mice with the suppressed genes to select for *recovery of function.*)

Fitting the ICR-only model is simple, and can be done using a variant of the above approach:

```r
# First ICR effect - total interaction
wtcicr <- rem.dyad(WTCPoliceCalls, n = 37, effects = c("CovInt"),
   covar = list(CovInt = WTCPoliceIsICR), hessian = TRUE)
```

```
Prepping edgelist.
Checking/prepping covariates.
Computing preliminary statistics
Fitting model
Obtaining goodness-of-fit statistics
```

```r
summary(wtcicr)
```

```
Relational Event Model (Ordinal Likelihood)

         Estimate  Std.Err Z value  Pr(>|z|)
CovInt.1 2.104464 0.069817  30.142 < 2.2e-16 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Null deviance: 6921.048 on 481 degrees of freedom
Residual deviance: 6193.998 on 480 degrees of freedom
    Chi-square: 727.0499 on 1 degrees of freedom, asymptotic p-value 0
AIC: 6195.998 AICC: 6196.007 BIC: 6200.174
```

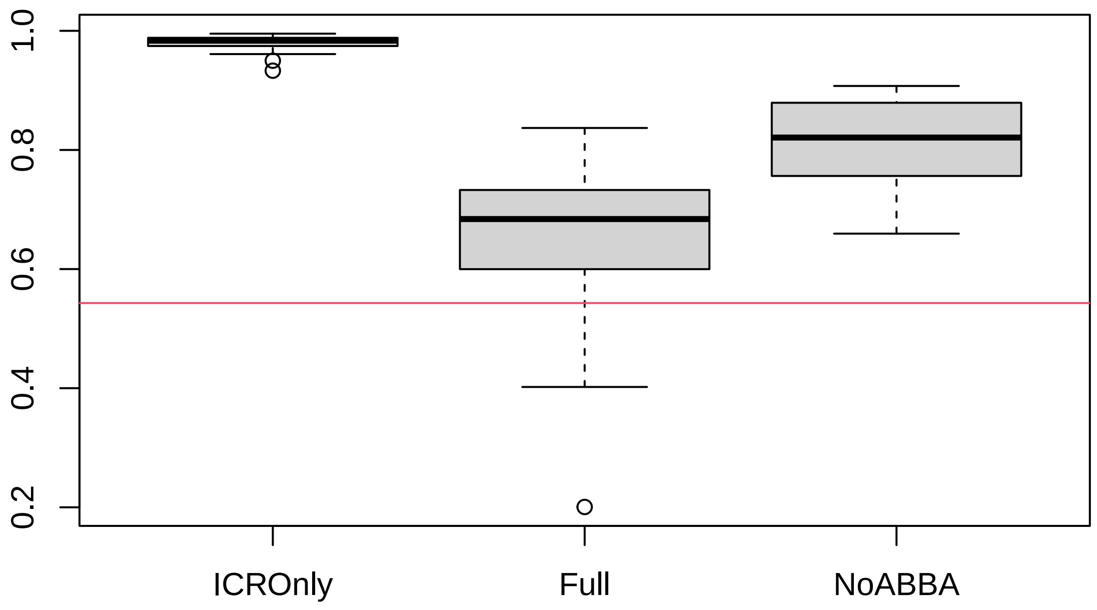
The effect is large, though we note that the BIC is large compared to our selected model. This is clearly a crude model, but that's fine: we are not interested in it as a general proxy for out social process, but as a "gendanken explanation" - the hypothetical best model we can fit from a single driving mechanism.

For our knock-out model, we remove the turn-taking effect (the AB-BA P-shift). Turn-taking is the most central element of radio SOP, and at first blush has little to do with hub formation. However, turn-taking has the effect of keeping conversations among the same partners, "piling up" communication on a small number of individuals. In concert with other effects (e.g., preferential attachment), this could conceivably lead to the formation of *emergent* hub roles not associated with ICR status. Alternatively, any impact of turn-taking could simply be to amplify differences in centrality between ICRs and other actors. If indeed AB-BA shift effects drive emergent hub formation *per se*, knocking them out should enhance the centrality difference between ICRs and others relative to control; by contrast, if AB-BA shifts in the empirically calibrated system are "amplifying" ICR effects, knocking them out should reduce them relative to the baseline control. Finally, our ICR-only model acts as a type of positive control: since, by design, ICR effects are the only term in the model, they represent what we might expect in a world in which communication is based entirely on ICR-status.

We simulate all three models using the `simulate` command, using the correlation between ICR status and betweenness in the time-aggregated communication network as our outcome measure. Our simulated trajectories are by default as long as the original trajectories to which the models were fit (but are simulated from "time 0"). Because the process is somewhat slow, we limit ourselves here to 25 replications.

```r
set.seed(13)
reps <- 25  #Number of replicate series to take
kocoef <- wtcfit$coef  #Knock-out coefs
kocoef["PSAB-BA"] <- 0
ICRBetCor <- matrix(nrow = reps, ncol = 3)
sim <- mclapply(1:reps, function(i) {
   # Run our simulation in parallel, if possible
   set.seed(i + ceiling(runif(1, 0, 1e+09)))  #Set a seed for this instance
   print(i)  #A progress indicator
   outvec <- vector()  #Our output, to be returned
   simwtc <- simulate(wtcicr, covar = list(CovInt = WTCPoliceIsICR))  #ICR only
   outvec[1] <- cor(betweenness(as.sociomatrix.eventlist(simwtc,
      37)), WTCPoliceIsICR)
   simwtc <- simulate(wtcfit, covar = list(CovInt = WTCPoliceIsICR))  #Final
   outvec[2] <- cor(betweenness(as.sociomatrix.eventlist(simwtc,
      37)), WTCPoliceIsICR)
   simwtc <- simulate(wtcfit, covar = list(CovInt = WTCPoliceIsICR),
      coef = kocoef)  #Knockout
   outvec[3] <- cor(betweenness(as.sociomatrix.eventlist(simwtc,
      37)), WTCPoliceIsICR)
   outvec  #Return the output
}, mc.set.seed = FALSE, mc.cores = mc.cores)
ICRBetCor <- t(sapply(sim, "["))  #Extract the results
```

```r
# Visualize the results
boxplot(ICRBetCor, names = c("ICROnly", "Full", "NoABBA"))
abline(h = cor(betweenness(as.sociomatrix.eventlist(WTCPoliceCalls,
    37)), WTCPoliceIsICR), col = 2)
```



```
# Verify that both means are indeed above the control
t.test(x = ICRBetCor[, 1], y = ICRBetCor[, 2])  #ICR only vs. control
```

```

    Welch Two Sample t-test

data:  ICRBetCor[, 1] and ICRBetCor[, 2]
t = 11.898, df = 24.515, p-value = 1.122e-11
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.2681107 0.3804973
sample estimates:
mean of x mean of y
 0.979405  0.655101
```

```
t.test(x = ICRBetCor[, 3], y = ICRBetCor[, 2])  #Knockout vs. control
```

```

    Welch Two Sample t-test

data:  ICRBetCor[, 3] and ICRBetCor[, 2]
t = 5.1086, df = 38.334, p-value = 9.256e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.09650456 0.22312955
sample estimates:
mean of x mean of y
```

```
 0.814918  0.655101
```

We can see here that (perhaps unsurprisingly) the ICR-only model overstates the relationship between occupying an ICR and having high betweenness; our full model does much better, generally producing realizations that cover the observed data (though, with only a few replicates, you may find that it sometimes doesn't!). What happens when we "turn off" the AB-BA shift? It turns out that this greatly increases the relative betweenness of ICRs, telling us that the AB-BA shifts are helping to play a role in keeping ICRs from inappropriately dominating the network. Why should turn taking matter here? The short answer is that turn-taking effects create opportunities for non-ICR responders to gain airtime, and end up as emergent coordinators. Taking out the AB-BA effect reduces emergent coordination, which in turn increases the relative centrality of the few individuals in institutionalized coordinative roles. This would have been very difficult to infer from simply examining trajectories, but can be easily seen using computational experiments.

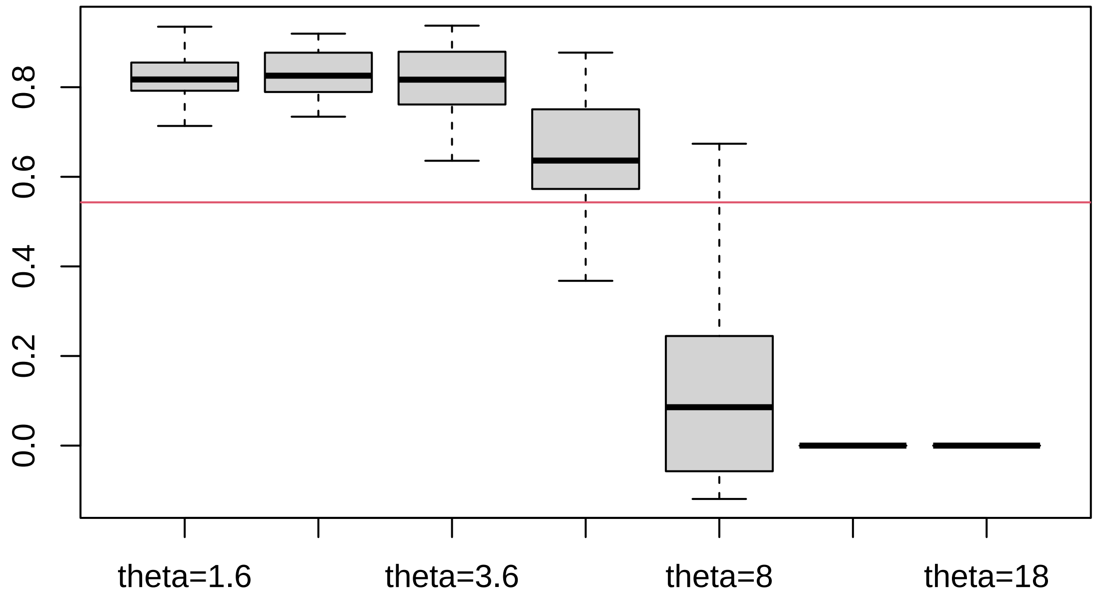
3.2 Knocking Turn-taking Up and Down

Just as we can learn a lot about the function of a mechanism by turning it on or off, we can also learn a lot by dialing it up or down. Such "knock-up/knock-down" experiments not only give us more quantitative insight into what happens when a mechanism is weakened, but also let us probe what happens when a mechanism is *strengthened*.

As an illustration, let us return to our above knock-out experiment. In that experiment, we removed the AB-BA P-shift effect (turn-taking), and saw that this enhanced the relative centrality of ICRs (putatively by inhibiting the rise of emergent coordinators). If this is so, does further enhancement of turn-taking result in even more decoupling of the ICR/coordinator association? And does weakening the effect have a smooth impact on the relationship, or is there some critical point at which turn-taking reciprocity is strong enough to override the impact of pre-existing roles? To probe this question, we take our observed AB-BA coefficient, and modify it both up and down, simulating interaction sequences in each case to see how turn-taking relates to our outcome of interest.

```
set.seed(13)
reps <- 25  #Number of replicate series to take
eff <- wtcfit$coef["PSAB-BA"] * 1.5^(-3:3)  #AB-BA effects to examine
sim <- mclapply(1:reps, function(j) {
   # Simulate over replications
   set.seed(j + ceiling(runif(1, 0, 1e+09)))  #Set a seed for this instance
   outv <- vector()  #Local output vector
   for (i in 1:length(eff)) {
      # Walk through conditions
      cat("Working on replication", j, "on effect", eff[i],
         "\n")
      kcoef <- wtcfit$coef  #Knock-up/down coefs
      kcoef["PSAB-BA"] <- eff[i]  #Substitute the current effect strength
      # Simulate from the target
      simwtc <- simulate(wtcfit, covar = list(CovInt = WTCPoliceIsICR),
         coef = kcoef)
      outv[i] <- cor(betweenness(as.sociomatrix.eventlist(simwtc,
         37)), WTCPoliceIsICR)
   }
   outv  #Return the result
}, mc.set.seed = FALSE, mc.cores = mc.cores)
KUDBetCor <- t(sapply(sim, "["))  #Extract the results
KUDBetCor[is.na(KUDBetCor)] <- 0  #Consider no variance = no relationship
```

```
# Visualize the results
boxplot(KUDBetCor, names = paste0("theta=", round(eff, 1)))
abline(h = cor(betweenness(as.sociomatrix.eventlist(WTCPoliceCalls,
    37)), WTCPoliceIsICR), col = 2)
```



Depending on your settings, you may see warnings arising from the fact that betweenness in some cases has no variance: what is happening (as inspection of the trajectories reveals) is that a sufficiently high AB-BA effect leads to a transcript composed of a single conversation between two individuals; since there is no brokerage in such a network (and no coordination), betweenness is zero for all individuals. We take the correlation to be zero here, since there is no meaningful association between coordination and ICR status in these networks.

Overall, what we find is a very striking pattern: the relationship between betweenness and ICR status is high near and below the observed AB-BE effect strength (increasing as the effect is weakened), but collapses catastrophically when the effect exceeds a threshold not far beyond the observed value. As noted, these networks become "hyper-centralized," with turn-taking norms becoming so strong that only a handful of individuals ever communicate. Such networks have no coordination at all, and are obviously quite pathological.

What do we learn from this? On the low-θ side, our knock-down experiment suggests that weakening turn-taking norms leads smoothly to a more ICR-dominated world (plausibly by reducing the supply of emergent coordinators); cutting the observed effect strength in half seems adequate to saturate in this direction. On the knock-up side, further increases in the strength of turn-taking quickly lead to catastrophic concentration of communication volume in small numbers of dyads, resulting in networks that are both implausible and non-functional. From this, we may glean that the observed strength of turn-taking reciprocity norms are about as strong as they can be without deleterious effects: we could imagine them to be stronger, but not much stronger. It is interesting to conjecture that the organizational routines embedded in radio SOP are thus approximately optimized to push this behavior close to its limit (e.g., to enhance accuracy by extensive confirmation of communications), though we cannot address that with this data. But we can certainly see that a plausible alternative model for this system could have lower levels of turn-taking, but not much higher (at least, not without adjusting other mechanisms to compensate).

Section 4. Knocking In Hypothetical Mechanisms

In addition to removing effects, we can also ask what happens if new effects are introduced to an empirically calibrated model. These "knock-in" experiments can provide additional evidence for how a specific mechanism works, and can also be useful for assessing hypothetical interventions.

4.1 Knocking In Gender Effects On Initiation Rate

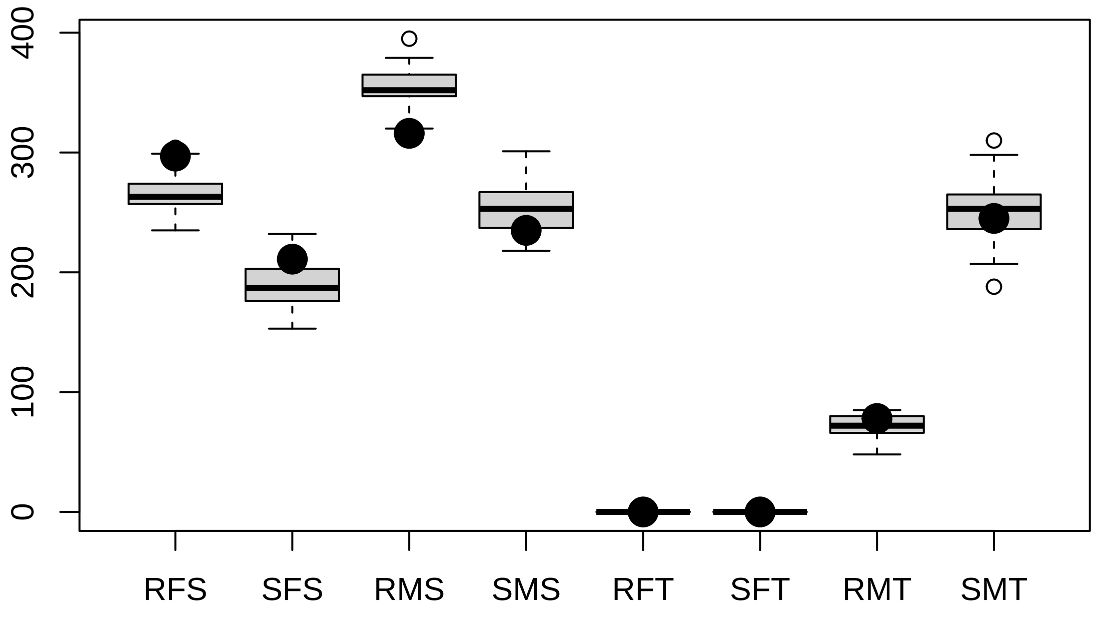
For instance, let us consider the McFarland classroom model. As noted above, participant gender information is available, but a gender effect was not found in the final model; controlling for teacher roles and endogenous dynamics, we do not observe a net tendency for male and female classroom participants to behave differently. We may consider, however, what impact such a difference would be expected to have on a classroom like the one observed, *were it present.* Naively, one might expect a simple, linear effect: more propensity to interact produces proportionately more interaction. However, the presence of endogenous social dynamics changes the picture, possibly amplifying or suppressing these effects. There may also be emergent effects, e.g. in teacher-student interaction, arising from differences in student communication. While we could imagine probing these questions with an entirely *de novo* model, starting with an empirically calibrated baseline allows us to probe the impact of hypothetical mechanisms within a more realistic setting.

To probe this question, we must first decide what about the system we want to measure. Here, we'll keep it simple, counting the number of events (communication volume) involving each individual, cross-classified by whether ego is the sender/reciever, ego's gender, and ego's classroom role (teacher/student). Note that we would expect these counts to be affected by the demographics of the classroom (*ceteris paribus*); however, we will be comparing them across conditions holding the classroom constant, so this is not an immediate concern. (If we were comparing across classrooms, however, we would probably want to normalize them.) To simplify analysis, it is helpful to create a function to tabulate the event data in this way. Here's one way to do it:

```
#Create a convenience function to tabulate communication by role and gender
classTab<-function(el){
  if(is.data.frame(el))    #If a data.frame, coerce to matrix form
    el<-as.matrix(el)
  if(is.list(el))          #If a list, run this on every entry
    return(t(sapply(el,classTab)))
  #If still here, tabulate the data, label, and return
  v<-as.vector(table(rep(as.factor(c("Snd","Rec")),each=NROW(el)), as.factor(c("Male","Female"))[1+Clas
  names(v)<-c("RFS","SFS","RMS","SMS","RFT","SFT","RMT","SMT")
  v
}

#Demonstrate on the observed data
obsvol<-classTab(Class[-NROW(Class),])     #Last row is the observation terminus
```

Observe that the function returns a vector of counts, with category succinctly described by a three-letter code ((R)eceiver/(S)ender; (F)emale/(M)ale; and (S)tudent/(T)eacher). Because both instructors are male, there are no RFT and SFT events - we remove these categories from our analysis below. Our interest will then be in comparing these counts in models where gender differences in sending rates have been knocked in, to the baseline control. Incidentally, while it is still sometimes possible to draw useful conclusions about *differences* in outcome behavior when the baseline model does not match the data (a common situation e.g. in computational chemistry), it is a good idea to check. Let's compare the control model to the data on our statistics of interest, and verify that it can do a credible job of reproducing this feature of the data.

```
set.seed(13)
# Create 25 replicate simulations of the class
reps <- 25
classsim <- mclapply(1:reps, function(i) {
   set.seed(i + ceiling(runif(1, 0, 1e+09)))  #Set a seed for this instance
   simulate(classfit, covar = list(CovSnd = cbind(ClassIntercept,
      ClassIsTeacher), CovRec = ClassIsTeacher))
}, mc.set.seed = FALSE, mc.cores = mc.cores)
simvol <- classTab(classsim)  #Obtain the communication volumes

boxplot(simvol)  #Plot the distributions
points(obsvol, pch = 19, cex = 2)  #Are we within the range?
```



We can see that the observed cases are within the range of the simulations, and the overall pattern is preserved. Thus, we regard the model as adequate for our target of interest.

To perform our knock-in study, we now want to take our baseline control, and add a new effect for the propensity to initiate interaction, by gender. There are many ways to parameterize such an effect, but it is helpful to do so in a way that is both easy to interpret, and that does not radically alter other aspects of the model (e.g., the mean propensity of students to initiate communication). Here, we choose to add a sending covariate whose value is 0.5 for female students, and -0.5 for male students (i.e., a *contrast*); such a term creates a net difference of θ between genders, without creating a net shift between groups. Note, though, that this will still create a net shift in the average sending rate in the student *population*, unless the gender balance is exactly even. In this network, approximately half of the students are female (8 female students to 10 male students), so this shift will be fairly small. Since reparameterizing to make it exactly zero would make the term asymmetric and harder to interpret at the individual level, we leave it as-is - but such unintended consequences are something to keep in mind when designing and interpreting *in silico* experiments.

Now, how does our experiment work? We vary the strength and direction of our hypothesized gender difference, at each level simulating a sample of replicate sequences and obtaining the communication volumes; comparing these allows us to see how the hypothesized effect changes the behavior of the system as a whole. To add the effect to our base model (`classfit`), we can simply add the additional covariate, and use the `coef` argument of the `simulate` function to pass a coefficient vector that combines the original coefficients and the coefficient or the new term. (Note that we have to be sure to add the term in the right place! `rem.dyad` orders its effects in a specific way, so looking at the model object will tell you where you need to insert the coefficient of interest.) We accumulate the mean communication volumes using `classTab` at each level of our treatment (the gender effect), allowing us to see how the former changes with the latter. This may be accomplished using a simple `R` script, like so:

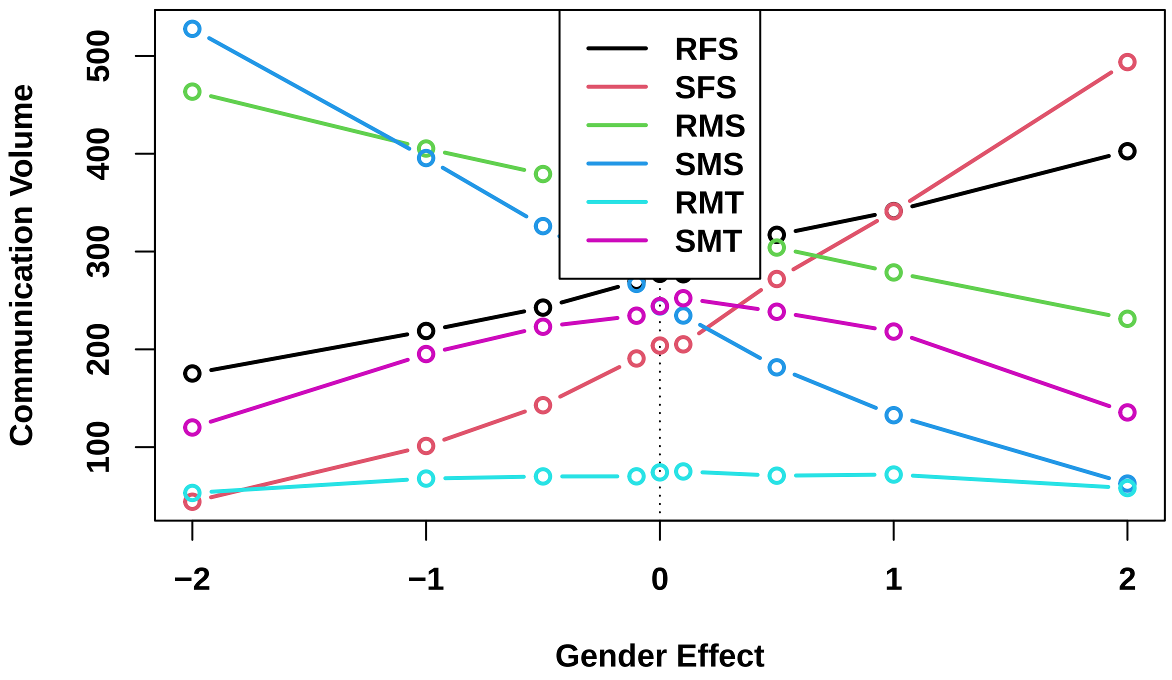
```
set.seed(13)  #Set seed for replication
ClassFMS <- (!ClassIsTeacher) * (ClassIsFemale - 0.5)  #Create the F/M contrast term
co <- c(classfit$coef[1:4], 0, classfit$coef[5:8])  #Set up the base coefficients
mcomvol <- vector()  #Mean communication volumes
eff <- c(-2, -1, -0.5, -0.1, 0, 0.1, 0.5, 1, 2)  #Effect strengths (treatments)
reps <- 25  #Replications at each level
for (i in 1:length(eff)) {
   # Walk through the levels
   cat("Working on eff =", eff[i], "\n")
   co[5] <- eff[i]  #Set the gender effect
   sim <- mclapply(1:reps, function(j) {
      # Accumulate replicates
      set.seed(j + ceiling(runif(1, 0, 1e+09)))  #Set a seed for this instance
      el <- simulate(classfit, covar = list(CovSnd = cbind(ClassIntercept,
         ClassIsTeacher, ClassFMS), CovRec = ClassIsTeacher),
         coef = co)
      classTab(el)
   }, mc.set.seed = FALSE, mc.cores = mc.cores)
   rcomvol <- t(sapply(sim, "["))
   mcomvol <- rbind(mcomvol, colMeans(rcomvol))  #Accumulate the average
}
```

We now have our communication volumes, with gender effects knocked in! Observe that we also have our baseline control, which arises when our new coefficient is set to zero - thus, we retain a point of comparison. To see the results, let us see how our coefficient is related to the mean communication volumes.

```
mcomvol   #Print the mean communication volumes
```

```
         RFS    SFS    RMS    SMS RFT SFT   RMT    SMT
 [1,] 175.32  44.20 463.48 527.68   0   0 53.20 120.12
 [2,] 218.80 101.20 405.32 395.60   0   0 67.88 195.20
 [3,] 242.68 142.88 379.24 326.00   0   0 70.08 223.12
 [4,] 269.12 190.68 352.72 266.92   0   0 70.16 234.40
 [5,] 277.20 203.84 340.56 243.88   0   0 74.24 244.28
 [6,] 276.80 205.12 340.04 234.60   0   0 75.16 252.28
 [7,] 317.08 271.92 304.08 181.64   0   0 70.84 238.44
 [8,] 341.36 341.12 278.64 132.68   0   0 72.00 218.20
 [9,] 402.56 493.72 231.24  62.76   0   0 58.20 135.52
```

```
#Plot communication volumes by type and effect size
plot(0,0,type="n",xlim=range(eff),ylim=range(mcomvol[,-(5:6)]), xlab="Gender Effect", ylab="Communicati
abline(v=0,lty=3)           #Mark the control case
for(i in 1:6)               #Plot each data series
  lines(eff,mcomvol[,-(5:6)][,i],type="b",col=i,lwd=2)
legend("top",lwd=2,col=1:6,legend=colnames(mcomvol)[-(5:6)],text.font=2,bg="white")
```



The mean communication volumes show a number of striking patterns. As expected, student sending volume by gender (SFS, SMS) scale with the gender effect - increasing the propensity to initiate communication does drive volume accordingly. However, we also see substantial (but somewhat flatter) changes in student receipt by gender (RFS, RMS), despite the lack of any effect for homophily or target selection. This is the result of reciprocity arising from turn-taking and recency mechanisms: when female (respectively male) students initiate interaction at a higher rate, they produce a disproportionate number of targets who are prone to reciprocate, leading to a higher incidence of incoming communication. Communication targets are thus indirectly biased by the initiation pattern, producing what at first blush looks like actively gendered target selection. Interestingly, we see a rather different spillover into teacher communication (RMT, SMT, recalling that both teachers are male). Unlike their students, the teachers show a largely symmetric, non-monotone pattern in both sending and receipt as a function of gender effect strength. Highest in the control condition, teacher communication declines when student propensity to initiate interaction shifts in either direction. Why? The obvious answer is that, as communication intensity becomes concentrated in a small number of increasingly loquacious students, who end up dominating the classroom by talking amongst themselves - conversational dynamics eventually begin to shut out even the teachers, who have fewer occasions to break in. Such emergent effects are difficult to anticipate by contemplating raw interaction propensities, but can be easily identified using simulation experiments.

Section 5. Probing Covariates, and Dose/Response Curves

The previous sections have focused on experiments that change mechanisms - adding them, removing them, or retuning them - as a way to understand model behavior. What about changing covariates? As covariates often tell us about the context in which action takes place, and/or the attributes of those engaging in it, changes of covariate are a powerful tool for studying the potential impact of differing circumstances on model behavior. This can be used to study potential generalizability of the model (does it continue to behave plausibly when we change the population?), to gain substantive insights via “what-if” scenario evaluation, or to plan interventions by examining the apparent sensitivity of the system to a possible change. Here, we will show some examples of this type of manipulation.

5.1 Changing Roles

In section 3, we showed how an *in silico* knock-out study could be used to gain insights into model behavior. Another useful strategy can be to simulate trajectories from a fitted model with alternative choices of covariates. For instance, what might we expect if we replaced the teachers in our classroom with students? This anarchic state of affairs can be probed by conditional simulation with a different set of covariates:

```
set.seed(13)
ClassSim <- simulate(classfit, covar = list(CovSnd = cbind(ClassIntercept,
   ClassIsTeacher), CovRec = ClassIsTeacher))
AnarchSim <- simulate(classfit, covar = list(CovSnd = cbind(ClassIntercept,
   rep(0, 20)), CovRec = rep(0, 20)))

head(AnarchSim)  #Examine the trajectory
```

```
          [,1] [,2] [,3]
[1,] 0.1001036   16    4
[2,] 0.1288086    9   17
[3,] 0.1718126   17    9
[4,] 0.3178658    9   17
[5,] 0.3293235    9   14
[6,] 0.4234096    9    8
```

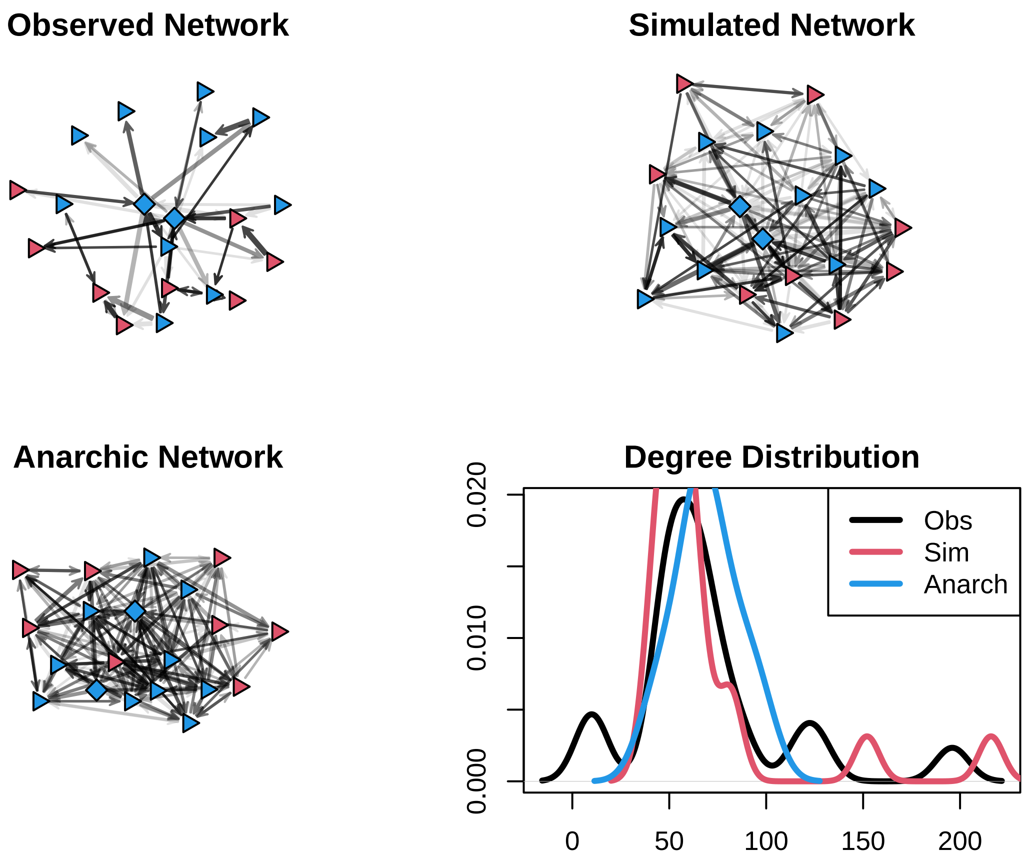
```
tail(AnarchSim)
```

```
           [,1] [,2] [,3]
[687,] 58.09008    1   16
[688,] 58.15661    1   17
[689,] 58.25372    2    3
[690,] 58.30740    7    8
[691,] 58.35169    8    7
[692,] 58.46455    7    8
```

The new trajectory is driven by the same mechanisms as the old, but the setting has now changed. What is the impact?

```
# Plot the network structure of the simulations, and the
# observed data
par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))
gplot(ClassNet, vertex.col = 4 - 2 * ClassIsFemale, vertex.sides = 3 +
   ClassIsTeacher, vertex.cex = 2, edge.lwd = ClassNet^0.75,
   main = "Observed Network", edge.col = rgb(0, 0, 0, (1 - 1/(1 +
      ClassNet))^3))
SimNet <- as.sociomatrix.eventlist(ClassSim, 20)  #Create a network from the fitted sim
gplot(SimNet, vertex.col = 4 - 2 * ClassIsFemale, vertex.sides = 3 +
   ClassIsTeacher, vertex.cex = 2, edge.lwd = SimNet^0.75, main = "Simulated Network",
   edge.col = rgb(0, 0, 0, (1 - 1/(1 + SimNet))^3))
AnarchNet <- as.sociomatrix.eventlist(AnarchSim, 20)  #Create a network from the anarchy sim
gplot(AnarchNet, vertex.col = 4 - 2 * ClassIsFemale, vertex.sides = 3 +
   ClassIsTeacher, vertex.cex = 2, edge.lwd = AnarchNet^0.75,
   main = "Anarchic Network", edge.col = rgb(0, 0, 0, (1 - 1/(1 +
      AnarchNet))^3))

# Plot the valued degree distributions
```

```
plot(density(degree(ClassNet), bw = "SJ"), lwd = 3, main = "Degree Distribution")
lines(density(degree(SimNet), bw = "SJ"), lwd = 3, col = 2)
lines(density(degree(AnarchNet), bw = "SJ"), lwd = 3, col = 4)
legend("topright", legend = c("Obs", "Sim", "Anarch"), lwd = 3,
   col = c(1, 2, 4))
```



Comparing the plots, we can see several things. First, we note some limitations of our fitted model: while it does relatively well at ensuring that the teachers are central, enduring that many of the strongest interactions are student-teacher interactions, creating a network in which strong interactions are localized to a fairly small number of (highly reciprocal) dyads, and reproducing the overall valued degree distribution, it also produces a large “halo” of weak side-interactions among the students that is not seen in the observed network. This suggests the potential for further model improvement.

Turning to our “anarchy in the classroom” model, however, we see that the effect of removing teachers is substantively reasonable. The nodes that were formerly teachers no longer have any particular significance, and are now well-mixed with their peers; likewise, without the teachers to focus attention, the network is as a whole much less centralized. Thus, the model does plausibly produce many of the effects one would expect to see from such a change in group composition. Such scenario-based probes can be a useful tool for assessing model behavior, as well as being of possible substantive interest in and of themselves.

5.2 Dose-Response Curves for Hypothetical Interventions

A particularly useful application of covariate based simulations is to predict the “dose-response” curves for hypothetical interventions: i.e., given a certain level of intervention, what is the corresponding response on the variable of interest? While *in silico* estimates of outcomes may be primitive, a reasonable model may give a far better estimate e.g. of how many individuals need to be reached with a putative intervention to have an effect than one would obtain from intuition or limited case experience. In a study design context, this can provide valuable information that may allow one to avoid accidentally sabotaging promising interventions due to insufficient dosing.

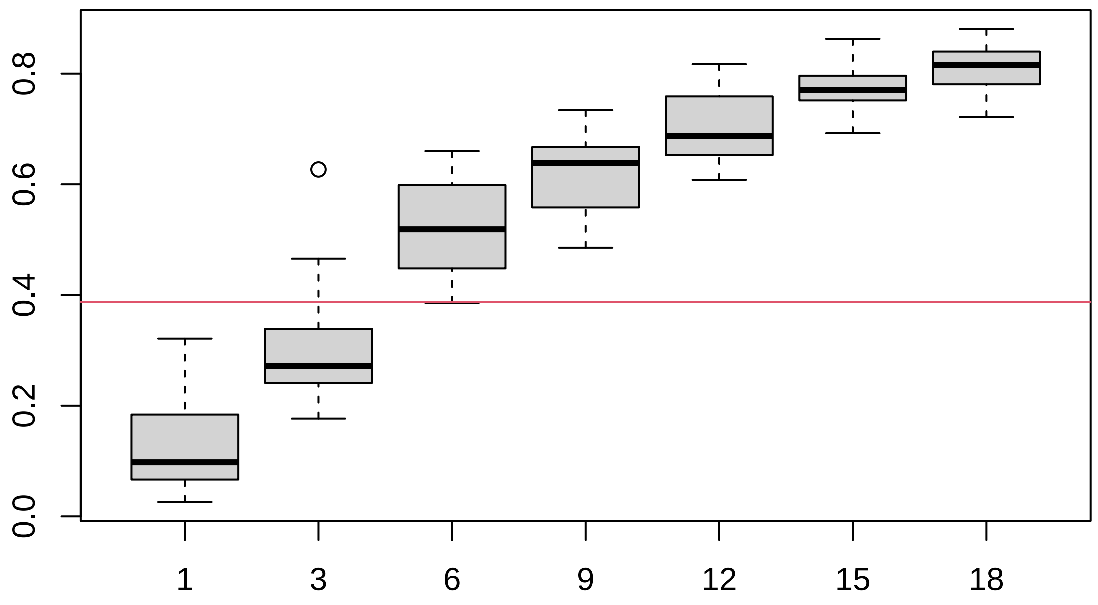
The approach here is straightforward: as before, we modify one or more covariates, in a manner that reflects the nature of the intended intervention. Frequently, it will be sensible to have a covariate reflecting the variable acted upon (or an indicator for being an intervention target), and to take the number of targets as the effective "dose." Evaluating outcome distributions by dose thus yields the desired information.

Here, we illustrate this notion with a simple (and somewhat contrived) case involving the WTC model. Communication during emergencies is costly and distracting, and coordinating the activities of others' is a difficult skill that is hard to execute under fire. This is, of course, one reason that response organizations have ICRs, who are both equipped and trained to handle this type of task. Consider a case in which one's objective is to have as much communication as possible handled by ICRs, to reduce the impact of coordination on the broader organization. Obviously, one can make all members of the organization ICRs, but this too is costly, and defeats the purpose; however, one might speculate that increasing the numbers of ICRs would improve performance, and be interested in determining the relationship between the number of ICRs in the network and the fraction of communication handled by ICRs.

Here, we examine this issue by computing a dose-response curve for the fraction of communicative events handled in some way by ICRs, as a function of the number of ICRs in the network. To implement this, we replace the `WTCPoliceIsICR` vector with a new vector having the desired number of ICRs (which we vary from 1 to 18). Since the model is otherwise permutation invariant, we can add the ICRs in any order, and indeed simply use the first k entries to place k ICRs.

```r
set.seed(13)
reps <- 25  #Number of replicate series to take
eff <- c(1/3, 1:6) * 3  #Number of ICRs to place
sim <- mclapply(1:reps, function(j) {
   # Simulate over replications
   set.seed(j + ceiling(runif(1, 0, 1e+09)))  #Set a seed for this instance
   outv <- vector()  #Local output vector
   for (i in 1:length(eff)) {
      # Walk through conditions
      cat("Working on replication", j, "on effect", eff[i],
         "\n")
      NewICR <- rep(1:0, times = c(eff[i], 37 - eff[i]))
      # Simulate from the target
      simwtc <- simulate(wtcfit, covar = list(CovInt = NewICR))
      d <- degree(as.sociomatrix.eventlist(simwtc, 37))
      outv[i] <- d %*% NewICR/sum(d)
   }
   outv  #Return the result
}, mc.set.seed = FALSE, mc.cores = mc.cores)
WTCICRShare <- t(sapply(sim, "["))  #Extract the results
```

```r
# Visualize the results
boxplot(WTCICRShare, names = eff)
abline(h = degree(as.sociomatrix.eventlist(WTCPoliceCalls, 37)) %*%
   WTCPoliceIsICR/(2 * NROW(WTCPoliceCalls)), col = 2)
```



The results first confirm that the observed value is within the range of the simulated behavior for the baseline control (there are 3 ICRs in the observed data set). Secondly, the simulations show a very orderly dose-response relationship, increasing rapidly but sublinearly until the network contains approximately 12 ICRs, and then tapering off. Although this is a large fraction of the network, the higher activity level of ICRs gives them a disproportionate share of the traffic; to place 60% of the traffic on ICRs, for instance, one needs only 6-9 ICRs (16-24%) of the network. With this information in hand, one would have a much more principled basis for deciding how many ICRs to create in a comparable group than e.g. simply sticking with what was done before, or making a guess about the number that would be sufficient. While this specific example is perhaps fanciful (it is unlikely that an organization would or should create ICRs on this basis), it illustrates an approach that can be profitably applied to a range of study design problems.

Section 6. Cutting In: Intervening in Trajectories

So far, we have considered only cases in which our trajectories begin *tabla rasa*, with the model simulating them from scratch. In other cases, however, we may be interested in what happens when we interrupt action mid-stream. The `simulate` method for `rem.dyad` supports such cases by allowing us to pass an event list containing history prior to the moment at which simulation begins - this is done using the `edgelist` argument, which takes as input a matrix of the same form as standard `rem.dyad` data, interpreted as the first `m` events in the sequence to be simulated. (Note, then, that if you want to simulate `m2` additional events, you need to set `nsim=m+m2` to get what you want.) It is also possible to redraw timing for these observed events (useful if they were observed without explicit timing information), or to keep the times and redraw the events themselves (a more exotic option). Many applications of this functionality are possible, including probes in which individuals are added/removed/changed mid-stream, specific events are added/removed, etc. Here, we show one illustrative example, in which we make one individual unavailable partway through an event sequence.

6.1 Taking a Teacher Out of the Classroom

In section 5.1, we considered a classroom where the teachers have been replaced by students. Here, we will consider a less extreme scenario: half-way through the class, a randomly selected teacher will be exogenously removed, and thus unavailable for interaction. What happens to the volumes of student/teacher and student/student interaction in this scenario, versus a baseline control where no change is made? We perform this experiment via three steps: first, we segment the original event list, selecting all observed events in the desired interval to use as a prefix for our intended sequence; second, we create sender and receiver covariates to mark the removed node (isolating them from further events); and then simulating conditional realizations of the altered sequence and model. We compare this to control simulations with the same structure, but in which the teacher is not removed.

We begin by defining our outcome function; this simply takes an input sequence and counts the number of events that involve a student and a teacher, versus two students (respectively).

```
#Create a new convenience function to count student/student and student/teacher events
classTab2<-function(el){
  if(is.data.frame(el))    #If a data.frame, coerce to matrix form
    el<-as.matrix(el)
  if(is.list(el))          #If a list, run this on every entry
    return(t(sapply(el,classTab)))
  #If still here, tabulate the data, label, and return
  v<-c(ST=sum(ClassIsTeacher[el[,2]]|ClassIsTeacher[el[,3]]), SS=sum((!ClassIsTeacher)[el[,2]]&(!ClassI
  v
}

#Demonstrate on the observed data
classTab2(Class[-NROW(Class),])        #Last row is observational terminus
```

```
 ST  SS
313 378
```

We next select the prefix. Since the class is about 50 minutes (and time is encoded in minutes), we take everything through the 25 minute mark.

```
ClassHalf <- Class[Class[, 1] < 25, ]  #Take all events in the first 25 min
NROW(ClassHalf)  #347 events
```

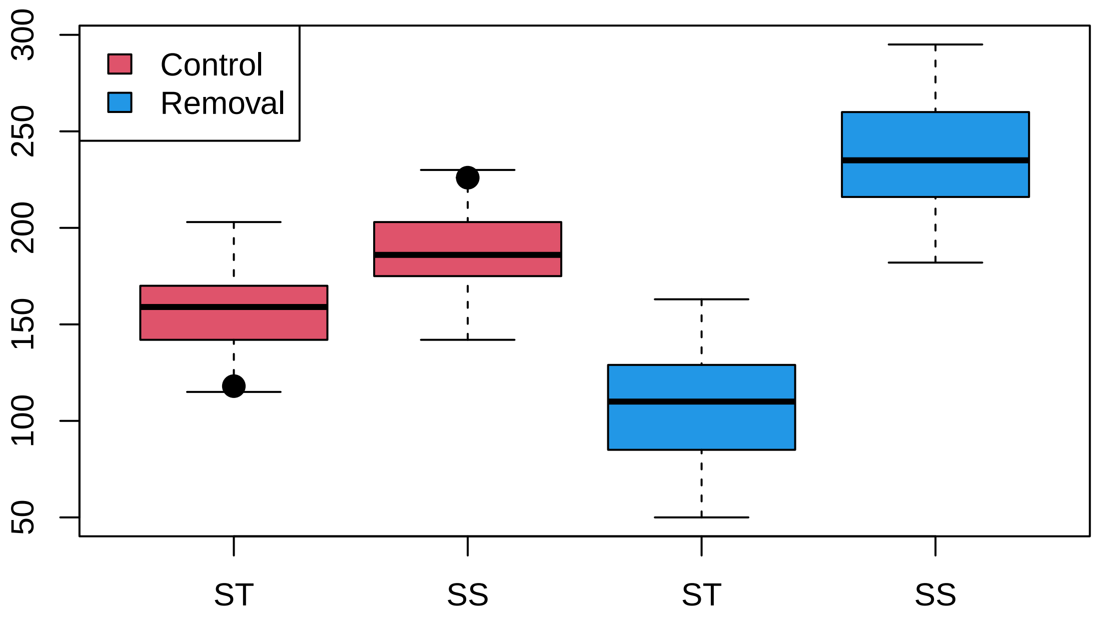
```
[1] 347
```

We will then simulate sequences of equal length to the original class, conditioning on the first half. Our code to do this task is as follows.

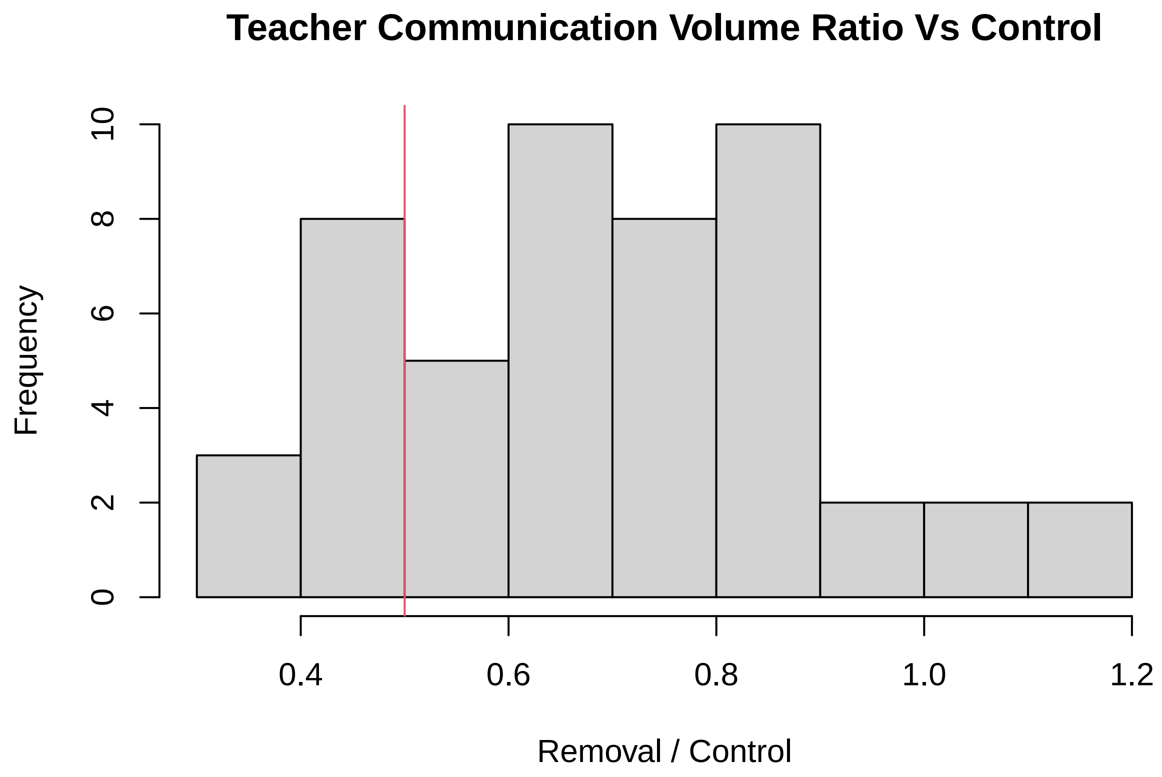
```
set.seed(13)  #Set seed for replication
reps <- 50  #Replications
# Create the coefficients, with space for new
# sender/receiver terms
co <- c(classfit$coef[1:4], 0, classfit$coef[5], 0, classfit$coef[6:8])
sim <- mclapply(1:reps, function(i) {
   # Accumulate replicates
   set.seed(i + ceiling(runif(1, 0, 1e+09)))  #Set a seed for this instance
   cat("Working on replication", i, "\n")
   Removed <- rep(FALSE, length(ClassIsTeacher))  #Pick a teacher to remove
   Removed[sample(which(ClassIsTeacher), 1)] <- TRUE
   # First, simulate with no removal
   co[c(5, 7)] <- 0
   el <- simulate(classfit, covar = list(CovSnd = cbind(ClassIntercept,
      ClassIsTeacher, Removed), CovRec = cbind(ClassIsTeacher,
      Removed)), coef = co, edgelist = ClassHalf)
   outv <- classTab2(el[-(1:NROW(ClassHalf)), ])  #Only consider second half
   # Now, simulate with removal
   co[c(5, 7)] <- -50  #Rate essentially 0
   el <- simulate(classfit, covar = list(CovSnd = cbind(ClassIntercept,
      ClassIsTeacher, Removed), CovRec = cbind(ClassIsTeacher,
      Removed)), coef = co, edgelist = ClassHalf)
   outv <- c(outv, classTab2(el[-(1:NROW(ClassHalf)), ]))
   outv
}, mc.set.seed = FALSE, mc.cores = mc.cores)
stcomvol <- t(sapply(sim, "["))
```

So what *is* the effect of removing a random teacher? What would we expect it to be? Naively, removing one of the teachers removes an interaction target, so we might expect the student/teacher volume to be cut in half. On the other hand, we could also imagine that the endogenous social mechanisms driving classroom interaction would lead interaction to be taken over by the remaining teacher, with little net loss. An extreme version of this hypothesis would posit that there is, for this system, a characteristic amount of student/teacher communication that is divvied up by the teachers, and when one is removed the other will take on the lost fraction. Which view is more accurate? Let us see.

```
# Visualize the results
boxplot(stcomvol, col = c(2, 2, 4, 4))
points(1:2, classTab2(Class[-c(1:NROW(ClassHalf), 692), ]), pch = 19,
   cex = 1.5)
legend("topleft", fill = c(2, 4), legend = c("Control", "Removal"))
```



```r
# How does teacher communication volume compare w/control?
hist(stcomvol[, 3]/stcomvol[, 1], xlab = "Removal / Control",
   main = "Teacher Communication Volume Ratio Vs Control")
abline(v = 0.5, col = 2)  #50% line
```



```r
mean(stcomvol[, 3]/stcomvol[, 1] > 0.5)  #Fraction > 50%
```

```
[1] 0.78
```

We can immediately see from the first plot that the strong "conserved teacher volume" thesis is wrong: removing a random teacher decreases student/teacher interaction (increasing student/student interaction in the bargain - but remember that we have fixed the number of events, so this is inevitable!). Does this imply that the naive, linear hypothesis is true? The second plot says otherwise: when we compare paired observations (treatment versus control), we see that student/teacher communication volume is usually greater than half of the control volume (about 75% of the time, in this case), with the median ratio being about 0.63. (The difference of the mean from the 50% ratio is also significant.) So what we see in this case is a bit of a compromise: removing a teacher from the classroom does markedly reduce student/teacher interactions, but there is some tendency of the remaining teacher to pick up some of the lost volume. The system thus adjusts to the disruption, but only partially.

Incidentally, our check on the adequacy of the model to reproduce the observed data (plotted points, first figure) suggests that the data is in the range of the model behavior, but the model appears to somewhat over-predict student/teacher interaction, and somewhat under-predict student-student interaction. While the model is probably good enough to be useful, we might be inclined to try to improve it if we were pursuing this outcome in detail. It is also useful here to remind ourselves that when we simulate our removal scenario, we are by design examining what would be expected to happen (under the model) *if the posited social mechanisms carried on as they were.* That can be a reasonable approximation, and even when it turns out not to be true it can still be a very interesting hypothesis to consider. However, we should not expect that it will hold in all situations: in some cases, an exogenous change to the social system will itself alter the mechanisms that drive behavior, possibly in substantial ways. Such limitations should be borne in mind in practical applications, and other empirical checks applied where feasible.

Section 7. *De Novo* Simulation of REMs

We have seen how the `simulate` command can be used to simulate draws from fitted `rem.dyad` objects, and even how these may be modified by switching coefficients or covariates for particular purposes. What if we want to create a *de novo* simulation? This can also be done, using `rem.dyad` to create a *model skeleton* that can subsequently be used for simulation.

7.1 Creating a Model Skeleton

To set up a REM for simulation, we need to create an object that records the system size (i.e., number of vertices), effects involved, and other critical information. When we fit models using 'rem.dyad'', this information was encoded in the model object. In the *de novo* case, we use the same approach - except that we simply omit the data!

To see how this is done, let's consider an example. Let us say that we want to create a model for a 25-node REM with a baseline intercept, an AB-BA P-shift, and a recency effect of sending on future sending (`RSndSnd`). We then proceed by creating a model just as we would normally, but with `NULL` where the data should be:

```
ModInt <- rep(1, 25)
modskel <- rem.dyad(NULL, n = 25, effects = c("CovSnd", "PSAB-BA",
   "RSndSnd"), covar = list(CovSnd = ModInt))
```

```
NULL edgelist passed to rem.dyad - creating model skeleton.
Checking/prepping covariates.
```

```
modskel
```

```
Relational Event Model
    Model skeleton (not fit)

Embedded coefficients:
      RSndSnd       CovSnd.1        PSAB-BA
 0.0006034927 -0.0009551033 -0.0002453042
```

Note that the model is correctly identified as a skeleton, with a reminder that it was not fit to data. It also comes equipped with "default" coefficients, but these are not very useful: if a seed coefficient is not passed, `rem.dyad` always initializes with perturbed coefficients near zero. If one knows what coefficients one wants to embed in the skeleton, one can set them using the `coef.seed` argument.

Note that none of the inferential or other arguments to `rem.dyad` are needed here, since no fitting is done. Perhaps less obviously, we do not need to set the `ordinal` variable, since all REM simulation is done in continuous time. (The resulting trajectories can, of course, be interpreted ordinally, if the pacing constant used was arbitrary.)

7.2 Simulating from the Model Skeleton

Simulation from the model skeleton is then performed just as simulation with fitted model objects, except that one needs to pass the number of draws to take (`nsim`, which was optional before) and `coef` (unless one already embedded the coefficients one wants in the model object). Be sure to enter your coefficients in the order stored in the skeleton, which may not be the order you initially specified the effects! Let's see how this works, using our example:

```
set.seed(1331)
modsim <- simulate(modskel, nsim = 100, coef = c(0.25, -1, 4),
   covar = list(CovSnd = ModInt))
head(modsim)  #See the trajectory
```

```
            [,1] [,2] [,3]
[1,] 0.003446229   20    2
[2,] 0.004616100    9    6
[3,] 0.005879407   25   17
[4,] 0.007751835   10   23
[5,] 0.009434835    9   20
[6,] 0.015124865   14   23
```

```
grecip(as.sociomatrix.eventlist(modsim, 25), measure = "edgewise")  #Relatively reciprocal
```

```
      Mut
0.2444444
```

Any number of events may be simulated in this way.

7.3 Simulation with Time-varying Covariates

Time-varying covariates must, by definition, be specified at each time step. `rem.dyad` understands several covariate formats (see `?rem.dyad`):

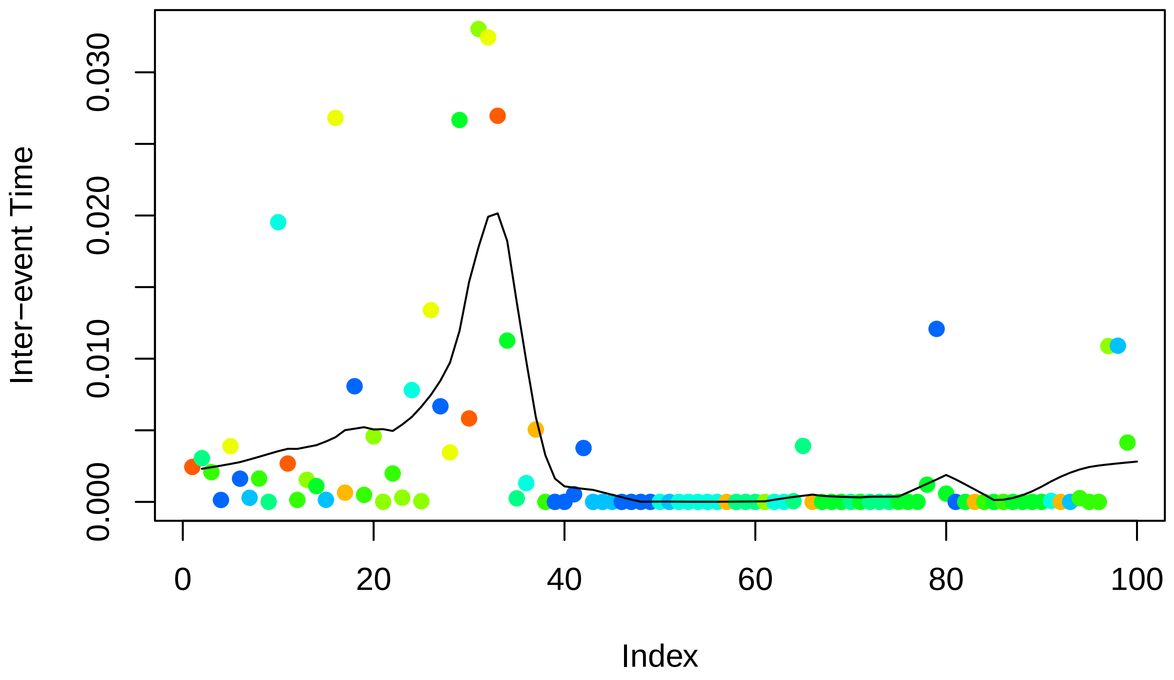
- Single covariate, time invariant: For `CovSnd`, `CovRec`, or `CovInt`, a vector or single-column matrix/array. For `CovEvent`, an `n` by `n` matrix or array.
- Multiple covariates, time invariant: For `CovSnd`, `CovRec`, or `CovInt`, a two-dimensional `n` by `p` matrix/array whose columns contain the respective covariates. For `CovEvent`, a `p` by `n` by `n` array, whose first dimension indexes the covariate matrices.
- Single or multiple covariates, time varying: For `CovSnd`, `CovRec`, or `CovInt`, an `m` by `n` by `p` array whose respective dimensions index time (i.e., event number), covariate, and actor. For `CovEvent`, a `m` by `p` by `n` by `n` array, whose dimensions are analogous to the previous case.

Thus, in the time-varying case, the dimensions of the covariate object must be consistent with `nsim`. Let's see a simple example, involving a 10-person group with an initial activity covariate that decays with time. We will simulate for 100 time steps, so need to create a 100 by 1 by 10 matrix to hold the covariate (the ith slice containing the covariate values "going into'' the ith event). When creating the skeleton, it is currently necessary to pass covariates as if they are static, since there are not yet multiple time points; the checks that are performed to ensure that the covariates are legal will object if too many time points are given. (This will probably change in the future.) The time-varying version is then passed to the simulator.

```
set.seed(1331)
# Set up the model
tcovar <- array(sweep(sapply(1:10, rep, 100), 1, 1/1.05^(0:99),
   "*"), dim = c(100, 10, 1))
SndInt <- rep(1, 10)
# Note that, in making the skeleton, we need to pass the
# covariates as if they are static - that's because the
# model doesn't contain time points yet.
modskel2 <- rem.dyad(NULL, n = 10, effects = c("CovSnd", "CovInt"),
   coef.seed = c(-1, 1), covar = list(CovSnd = SndInt, CovInt = tcovar[1,
      , 1]))
```

```
NULL edgelist passed to rem.dyad - creating model skeleton.
Checking/prepping covariates.
```

```r
# Simulate draws
modsim2 <- simulate(modskel2, nsim = 100, covar = list(CovSnd = SndInt,
   CovInt = tcovar))

# Note that dynamics slow down, and participation evens out
plot(diff(modsim2[, 1]), col = hsv(modsim2[, 2]/10 * 0.6), pch = 19,
   ylab = "Inter-event Time")
lines(supsmu(x = 2:100, y = diff(modsim2[, 1])))
```



On average, dynamics slow down, as we would expect, and more low-numbered (redder) vertices interact after the initial period.

References

Bender-deMoll, Skye, and Daniel A. McFarland. 2006. "The the Art and Science of Dynamic Network Visualization." *Journal of Social Structure* 7.

Butts, Carter T. 2008. "A Relational Event Framework for Social Action." *Sociological Methodology* 38 (1): 155–200.

Butts, Carter T., Alessandro Lomi, Tom A. B. Snijders, and Christoph Stadtfeld. 2023. "Relational Event Models in Network Science." *Network Science*. https://doi.org/10.1017/nws.2023.9.

Butts, Carter T., Miruna Petrescu-Prahova, and B. Remy Cross. 2007. "Responder Communication Networks in the World Trade Center Disaster: Implications for Modeling of Communication Within Emergency Settings." *Journal of Mathematical Sociology* 31 (2): 121–47.

DuBois, Christopher, Carter T. Butts, Daniel McFarland, and Padhraic Smyth. 2013. "Hierarchical Models for Relational Event Sequences." *Journal of Mathematical Psychology* 57 (6): 297–309.

Marcum, Christopher, and Carter T. Butts. 2015. "Constructing and Modifying Sequence Statistics for Relevent Using informR in R." *Journal of Statistical Software* 64 (1): 1–36. https://doi.org/10.18637/jss.

v064.i05.

Renshaw, Scott L., Selena M. Livas, Miruna G. Petrescu-Prahova, and Carter T. Butts. 2023. "Modeling Complex Interactions in a Disrupted Environment: Relational Events in the WTC Response." *Network Science*. https://doi.org/10.1017/nws.2023.4.