Online Appendix: Statistical Inference for Multilayer Networks in Political Science*

Ted Hsuan Yun Chen†

Contents

*Chen, Ted Hsuan Yun. "Statistical Inference for Multilayer Networks in Political Science." *Political Science Research and Methods*.

†ted.hsuanyun.chen@gmail.com. Department of Computer Science, Aalto University; Faculty of Social Sciences, University of Helsinki.

A Companion R package: `multilayer.ergm`

`multilayer.ergm` is an R package that extends the exponential random graph model to multilayer networks. It does so by extending functionalities of the `ergm` package to provide a way to count local network configurations, or network motifs, that span more than one network layer.

The package lives at its Github page (`https://github.com/tedhchen/multilayer.ergm`) and can be installed by entering the following into the R console. This requires you to have the `devtools` package installed. If you are using the Windows operating system, you will also need the standalone Rtools.[1] The argument `build_opts` can be set to default (i.e. remove the argument when running the function) if you do not require the package tutorial.

```
devtools::install_github("tedhchen/multilayer.ergm",
                         build_opts = c("--no-resave-data", "--no-manual"))
```

The package tutorial can be accessed by entering the following into the R console. The tutorial contains a brief introduction of how to create multilayer networks from monoplex networks and how to fit an ERGM to the multilayer network.

```
vignette("policy_multiplex")
```

The package is subject to further development. Version 0.1.6 ensures reproduction of results from this paper. For updates, please visit the package Github repository.

[1] `https://cran.r-project.org/bin/windows/Rtools/`

B Global Conflict Network: Data and Methods

This section contains a description of the data and methods for the global conflict application.

B.1 Data, Measures, and Network Creation

This study requires two sets of data. First, conflict data is needed to construct the multilayer network. For this, I used dyadic militarized interstate dispute data (Maoz 2005) for interstate conflict ties, and conflict data from the Uppsala Conflict Data Program (Allansson, Melander and Themnér 2017) for civil conflict ties and nonstate conflict ties. Second, data on node and dyad characteristics are required for the ERGM specification. I obtained this set of data from the replication files of Gleditsch, Salehyan and Schultz (2008).

Most of the raw data exist in either time-stamped or annual summary format. Because I model the post-Cold War years (1989–1995) as a single time period, the first step in processing the data is to aggregate them for the period. My aggregation algorithms are summarized in Table B1.

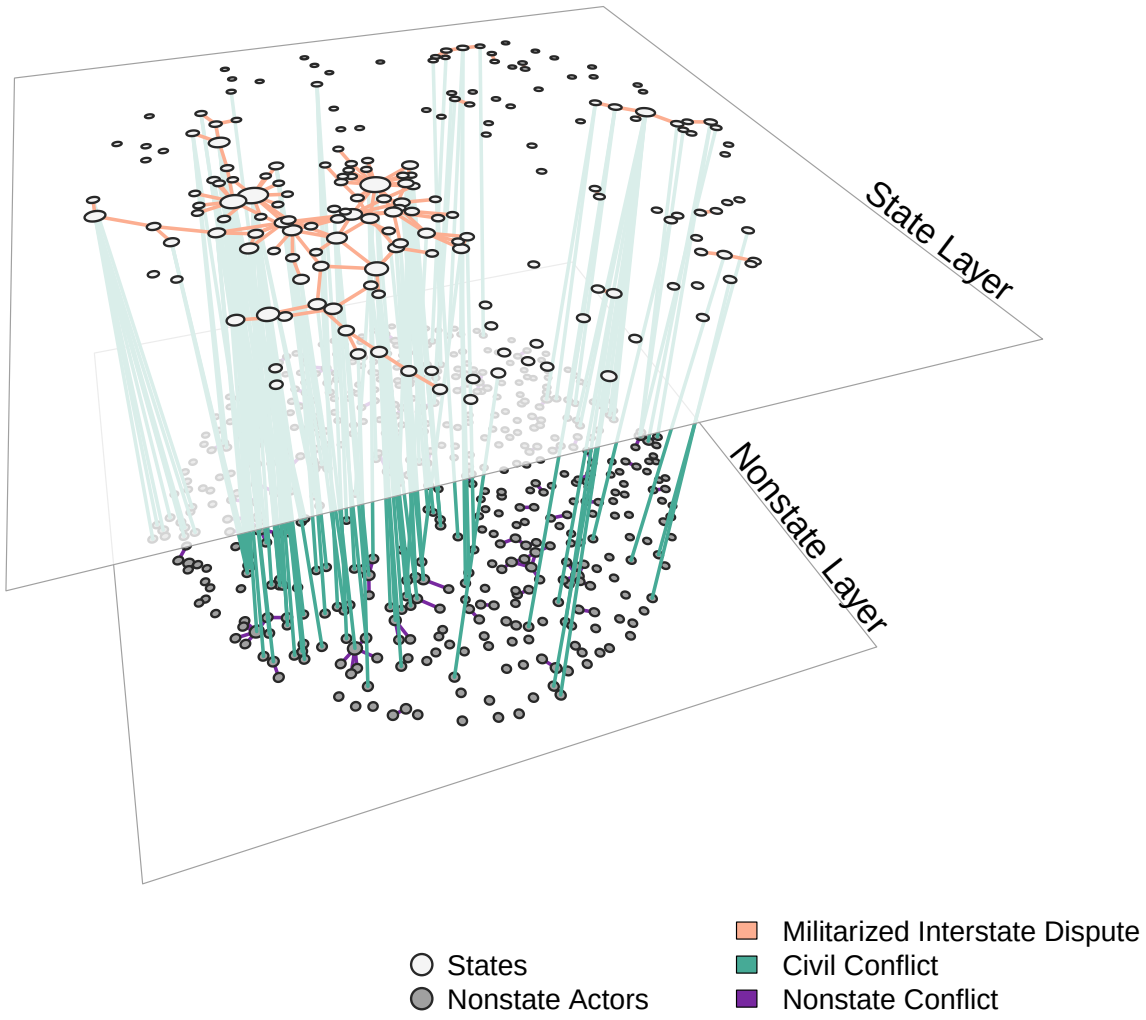
Table B1: Summary of Algorithm Used to Aggregate Data

Measures	Aggregation Algorithm
MID onset; civil conflict; nonstate conflict	Presence of at least one observation during period
Democracy*; joint democracy*	Held status for entire period
Transitional regime; political relevance; contiguity	Held status for at least one year during period.
Logged CINC; logged CINC ratio*	Median annual value during period

*These measures are lagged by one year for the entire period.

Next, to create the global conflict network, I first defined the set of nodes. All states that were part of the interstate system during the period are included, with the exception of Samoa, for which several variables are missing with no appropriate substitute. Nonstate actors are included if they were observed to be involved in any kind of organized conflict up to five years before and after the period under examination (1984-2000), based on data from the Uppsala Conflict Data Program (Allansson, Melander and Themnér 2017). Specifically, I used version 17.1 of the Dyadic Dataset, the Non-State Conflict Dataset, and the One-sided Violence Dataset. As in the Levantine conflict network, these nodes are organized by their statehood into two layers, state and nonstate. These two layers are then respectively populated with 189 states and 544 nonstate actors. As a node-colored network, the two layers yield three sets of ties based on the conflict data described above. This network is visualized in Figure B1.

Figure B1: Global Conflict Network, Post-Cold War Period



B.2 Modeling Steps

In this section, I describe the steps I take to arrive at my model of the global conflict network with cross-layer dependence. As in the ChemG application, I fit two ERGMs that differ by whether they include cross-layer dependence terms, and show that accounting for cross-layer dependence yields both substantive insights and a model that better fits the observed data. Whereas in the previous application I was able to use Leifeld and Schneider's (2012) model specification, extending it by adding cross-layer terms, the present application requires that I begin by finding a well-fitting model for each of the three sets of ties.

First, I specify the presence of civil conflict as an exogenous covariate for state nodes in

the MID submodel following Gleditsch, Salehyan and Schultz (2008). Specifically, engagement in civil conflict is specified as a nodal attribute that influences the likelihood of a state to form MID ties on the network. In network models, this is commonly referred to as an "activity" effect. Similar to the MID submodel, the submodels for civil and nonstate conflict networks include activity terms for whether nodes are presently engaged in the type of conflict not modeled as network ties. For additional exogenous node and dyad covariates, I draw on the set of variables from Gleditsch, Salehyan and Schultz (2008, Table 2, p. 491) as reference for the MID submodel. The submodels for the civil and nonstate conflict ties are sparse, as there is a general lack of data for nonstate actors. While there are efforts to rectify this (e.g. Cunningham, Gleditsch and Salehyan 2013; Fjelde and Nilsson 2012), the actors contained in these data sets are restricted compared to the population of nodes on the present network. I therefore limit these models to include only attributes associated with states.

For each submodel, I also include a set of network dependence terms that do not operate across different types of ties. I focus primarily on network structures that capture the tendency for conflicts to cluster, as this remains a persistent question in conflict studies (Braithwaite 2005; Gleditsch 2007; Buhaug and Gleditsch 2008; Gleditsch, Salehyan and Schultz 2008; Gibler and Braithwaite 2013). Clustering in conflict studies commonly refers to the general tendency for actors in proximity to other actors involved in conflict to also become involved in conflict (Buhaug and Gleditsch 2008; Gleditsch 2007). The most localized form of a conflict cluster is when one actor is simultaneously engaged in conflict with two other actors. In network terms, this local structure is called the "two-star". When the count of two-stars on a network is specified as an ERGM term, it can be used to capture the tendency for ties to cluster around a small number of nodes (Koskinen and Daraganova 2013). Cranmer and Desmarais (2011) find this to be a generative feature of the MID conflict network. In many social networks however, the "popularity" of an actor is likely to eventually plateau as opposed to increase unconstrained. Alternatively put, while there will be clustering in the form of multiple conflicts sharing the same center node, each additional tie formed by this node becomes less likely than the previous one. This is an oft-observed tendency for social networks (Koskinen and Daraganova 2013), and is likely to hold for conflict networks as well, given that piling-on is subject to diminishing gains and geographical constraints.

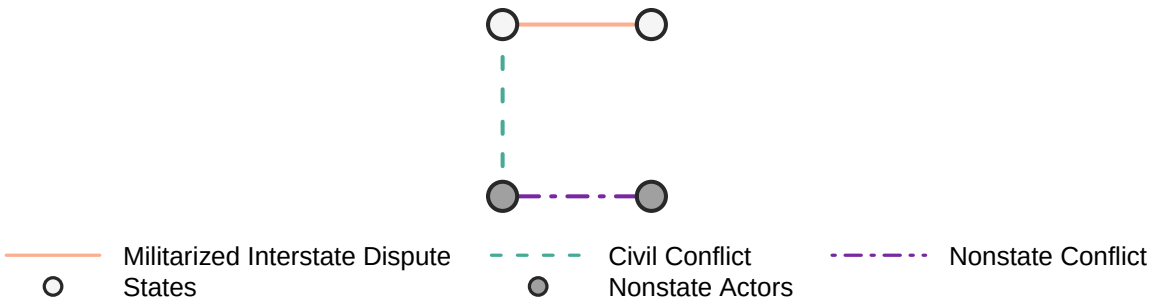
To account for this, I include the alternating k-star term, which is an extension to the two-star count that attenuates the tendency of tie-clustering as clustering increases (Snijders et al. 2006; Robins et al. 2007). In addition to its theoretical appeal, the alternating k-star terms improve model fit and reduce their tendency for degeneracy in the estimation procedure (Snijders et al. 2006). Interpretation of the estimated coefficients of these alternating k-star terms are similar to that of the two-star; a positive coefficient indicates clustering around a

relatively small number of high-degree nodes and a negative coefficient indicates a relatively equal distribution of conflict across the system.

The steps described above yield the layer independence model. Recall from the ChemG multiplex application that if the system is modeled without dependence across different sets of ties (i.e. different sets of ties are not constrained to have the same likelihood of being observed and there are no cross-layer dependence terms), estimated coefficients for such a model will be identical to the three sets of ties modeled in three separate models. As before, the layer independence model will fit the observed system poorly if interdependence exists across the three different conflict subsystems.

From here, I specify the cross-layer dependence model by replacing the exogenous conflict activity terms (e.g. civil war activity on the MID network) with two alternating k-star terms that are computed over two sets of ties at the same time. These are cross-layer dependence terms that respectively capture the tendency for MIDs and civil conflicts to cluster, and the tendency for civil and nonstate conflicts to cluster, both subject to plateauing in the same manner as within-layer alternating k-star terms (Wang, Robins and Matous 2016). As noted in the discussion of the Levantine conflict network, to specify one set of conflict as predictors of another requires assumptions about causal direction that are rarely justified in conflict studies. The cross-layer clustering terms allow me to model interdependence as network effects, thereby relaxing these assumptions.

Figure B2: Local Network Structure for Three-way Clustering



Finally, to account for the potential for all three types of conflicts to cluster together, a generative feature that cannot be captured without the use of cross-layer dependence terms, I introduce a three-way clustering term computed over all three tie sets. This particular clustering term captures the tendency for three-way clustering by counting the number of edges incident to at least one cross-layer three path, which is shown in Figure B2. The term I introduce here differs from the conventional approach of modeling higher-ordered clustering, which is by counting cross-layer three path structures (Wang et al. 2013). Similar

to the alternating k-star terms described above, this term downweights additional clustering ties beyond the first, and avoids the tendency for model degeneracy associated with more conventional structure counts (Snijders et al. 2006).

C Global Conflict Network Model Fit Comparison

In this section, I present results demonstrating that the cross-layer dependence model better fits the observed network compared to the layer independence model. Given the substantive focus of this application, I evaluate how closely simulated networks from each model resemble how interstate conflict dyads are distributed base on how they cluster with civil and nonstate conflicts. I refer to these clustering scenarios as the sub-state level strategic environment facing states that are involved in interstate conflict. These strategic environments are stylistically illustrated in Figure C1, and Table C1 provides one example of each type during the post-Cold War period.

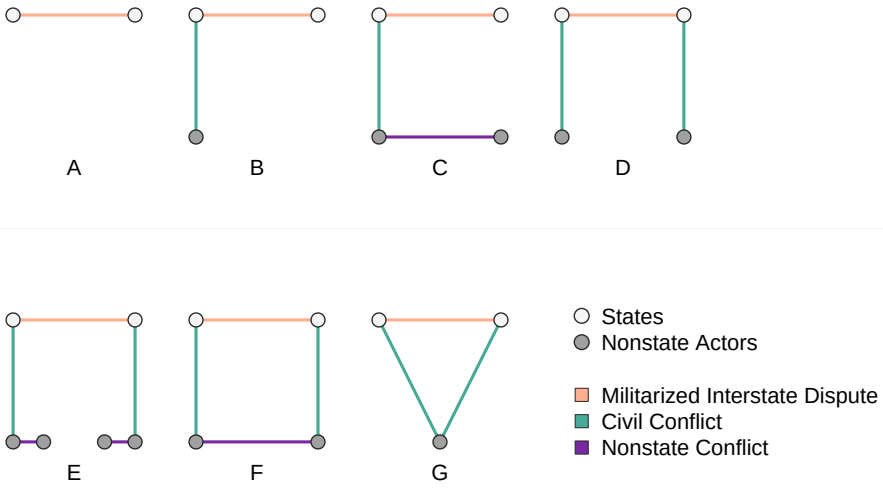
For each simulated network, I take the set of MID ties on the multilayer network, partition them into subsets according to the strategic environment under which they occur, and count the size of these subsets. For example, for an interstate conflict dyad to be in environment D, both states have to be involved in at least one civil conflict with none their civil conflict partners further fighting nonstate wars. When partitioning interstate conflict ties into these strategic environments, I treat the presence of multiple ties the same as one tie, meaning that regardless of how many civil conflict partners two states have, as long as none of them are further engaged in nonstate conflict, the interstate dyad faces environment D. Finally, all conflict dyads that share civil conflict partners are classified as G regardless of whether these nonstate conflict partners are involved in other conflicts.

Table C1: Examples of Strategic Conflict Environments

	MID Dyad		Nonstate Actors	
A	North Korea	South Korea		
B	United States	Iran	Democratic Party of Iranian Kurdistan	
C	Ecuador	Peru	Túpac Amaru Revolutionary Movement	Shining Path
D	Colombia	Nicaragua	FARC	Contras
E	Israel	Lebanon	Hezbollah	Forces of Michel Aoun
F	Iraq	Turkey	Patriotic Union of Kurdistan	Kurdistan Worker's Party
G	Afghanistan	Russia	National Islamic Front of Afghanistan	

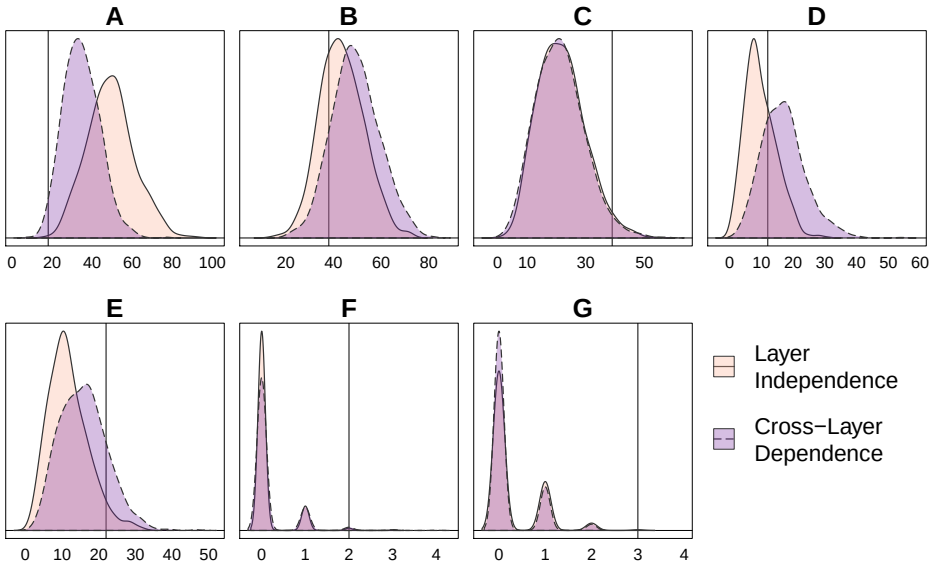
I simulate 1000 networks from each model and examine the resulting distributions, which

Figure C1: Census of Strategic Conflict Environments



are visualized in Figure C2. In each subfigure, the simulated distribution of counts of the given strategic environment are colored purple (dashed line) for the cross-layer dependence model and peach (solid line) for the layer independence model. The vertical black line is the value from the observed network. We should expect cross-layer dependence models to yield improvement in fit only if dependence exist and only if these dependences are modeled. Here, it is important to note that the three-way clustering term is the only additional information specified in the cross-layer dependence model that is not available in the layer independence model. The resulting distributions of simulated networks indicate that the inclusion of this additional piece of information either improves or maintains model fit across all strategic environments. Specifically, by including cross-layer dependence terms into the model, I obtain better fit for three of the seven strategic environments, which suggests that the model more closely captures the strategic considerations of states choosing to engage in conflict. In certain environments, both models fit poorly. This can likely be attributed to the sparsity of the submodels for civil and nonstate conflicts. These results read collectively indicate the value of cross-layer dependence models in improving model fit over models constrained to not account for cross-layer dependence.

Figure C2: Comparison of Model Fit for Different Strategic Environments



D Annotated Reproduction Code

This section contains the annotated code to reproduce results presented in the paper and online appendix. To ensure everything reproduces smoothly, please begin by running everything in D.1. The code in each substantive section (D.2 and D.3) can be run separately, but order within section is required. Four parallel processes are required to run the ERGM portions of the reproduction code.

D.1 Preparation

Begin by preparing the workspace and loading required packages. The `multilayer.ergm` package can be installed from its Github repository using the `devtools::install_github` function. The required data are lazily loaded by the package, so users do not have to explicitly load the data before using it. Note that the code forces installation of version 0.1.6. This ensures that updates to the package will not interfere with reproduction of results from this paper.

```
rm(list = ls())
setwd() # Set working directory (this is where the output files will be saved)

# Creating output folder
if(!dir.exists("./output")){dir.create("./output")}

# Loading packages
# Installs the "devtools" package if not already installed
if(!require("devtools")){install.packages("devtools")}

# Installs ergm package version 3.10.1
devtools::install_version("ergm", version = "3.10.1", upgrade = F,
                          repos = "http://cran.us.r-project.org")

# Installs the "multilayer.ergm" package from Github
devtools::install_github("tedhchen/multilayer.ergm@v0.1.6", upgrade = F)
library(multilayer.ergm)
```

D.2 ChemG Application

The ChemG application draws on Leifeld and Schneider (2012). Data directly obtained from their replication files and slightly cleaned.[2] The code in D.2 takes approximately 1 hour and

[2] http://hdl.handle.net/1902.1/17004

30 minutes to run using Amazon Web Service's C5 instances.

D.2.1 Data Preparation

```
rm(list = ls())
set.seed(804713)

# Organizing into multilayer structure.
# Political communication on layer 1; scientific communication on layer 2.
nw <- cbind(rbind(pol, diag(30)), rbind(diag(30), sci))

# Into network class
nw <- network(nw, directed = T)
nw%v%"layer.mem" <- c(rep(1, 30), rep(2, 30))
nw%v%"type" <- rep(types, 2)

# Constraining the off-diagonal matrix blocks
free <- matrix(c(rep(c(rep(1, 30), rep(0, 30)), 30), rep(c(rep(0, 30), rep(1, 30)), 30)),
               nrow = 60, byrow = T)
diag(free) <- 0
free <- network(free, directed = T)
```

D.2.2 Fitting ERGMs

```
# Layer independence model
mod.reduced <- ergm(nw
                    ~edgecov_layer("edges", layer = 1)
                    +gwesp_layer(decay = 0.1, fixed = T, layer = 1)
                    +gwdsp_layer(decay = 0.1, fixed = T, layer = 1)
                    +nodeifactor_layer("type", base = 2:5, layer = 1)
                    +nodeofactor_layer("type", base = 1:4, layer = 1)
                    +edgecov_layer(sci, layer = 1)
                    +edgecov_layer(committee, layer = 1)
                    +edgecov_layer(infrep, layer = 1)
                    +edgecov_layer(prefsim, layer = 1)
                    +edgecov_layer("edges", layer = 2)
                    +gwesp_layer(decay = 0.1, fixed = T, layer = 2)
                    +gwdsp_layer(decay = 0.1, fixed = T, layer = 2)
                    +nodeifactor_layer("type", base = 2:5, layer = 2)
                    +nodeofactor_layer("type", base = 1:4, layer = 2)
                    +edgecov_layer(pol, layer = 2)
```

```
                    +edgecov_layer(committee, layer = 2)
                    +edgecov_layer(infrep, layer = 2)
                    +edgecov_layer(prefsim, layer = 2)
                    +mutual(same="layer.mem", diff = T)
                    +nodemix(c("layer.mem", "type"), base = -c(12, 67)),
                    # Settings
                    eval.loglik = T, check.degeneracy = T, verbose = T,
                    control=control.ergm(seed = 6156713,
                                         MCMC.burnin = 20000,
                                         MCMC.samplesize = 20000,
                                         MCMC.interval = 2000,
                                         parallel = 4),
                    constraints = ~fixallbut(free))

# Model summary output (Table 1)
sink(file = "output/chemg_mod.reduced.summary.txt")
summary(mod.reduced)
sink()

# Model fit output
gof.reduced <- gof(mod.reduced, control = control.gof.ergm(nsim = 1000, seed = 1406018))
pdf("output/chemg_mod.reduced.gof.pdf", height = 8, width = 8)
plot(gof.reduced)
dev.off()

pdf("output/chemg_mod.reduced.mcmc.pdf", height = 8, width = 8)
mcmc.diagnostics(mod.reduced)
dev.off()

# Layer dependence model
mod.full <- ergm(nw
                 ~edgecov_layer("edges", layer = 1)
                 +gwesp_layer(decay = 0.1, fixed = T, layer = 1)
                 +gwdsp_layer(decay = 0.1, fixed = T, layer = 1)
                 +nodeifactor_layer("type", base = 2:5, layer = 1)
                 +nodeofactor_layer("type", base = 1:4, layer = 1)
                 +edgecov_layer(committee, layer = 1)
                 +edgecov_layer(infrep, layer = 1)
                 +edgecov_layer(prefsim, layer = 1)
                 +edgecov_layer("edges", layer = 2)
                 +gwesp_layer(decay = 0.1, fixed = T, layer = 2)
                 +gwdsp_layer(decay = 0.1, fixed = T, layer = 2)
                 +nodeifactor_layer("type", base = 2:5, layer = 2)
```

```
                  +nodeofactor_layer("type", base = 1:4, layer = 2)
                  +edgecov_layer(committee, layer = 2)
                  +edgecov_layer(infrep, layer = 2)
                  +edgecov_layer(prefsim, layer = 2)
                  +mutual(same="layer.mem", diff = T)
                  +nodemix(c("layer.mem", "type"), base = -c(12, 67))
                  +duplexdyad(type = letters[c(5 ,6, 7, 8)]),
                  # Settings
                  eval.loglik = T, check.degeneracy = T, verbose = T,
                  control = control.ergm(seed = 556914,
                                         MCMC.burnin = 20000,
                                         MCMC.samplesize = 20000,
                                         MCMC.interval = 2000,
                                         parallel = 4),
                  constraints = ~fixallbut(free))

# Model summary output (Table 1)
sink(file = "output/chemg_mod.full.summary.txt")
summary(mod.full)
sink()

# Model fit output
gof.full <- gof(mod.full, control = control.gof.ergm(nsim = 1000, seed = 1180906))
pdf("output/chemg_mod.full.gof.pdf", height = 8, width = 8)
plot(gof.full)
dev.off()

pdf("output/chemg_mod.full.mcmc.pdf", height = 8, width = 8)
mcmc.diagnostics(mod.full)
dev.off()

# Saving ERGM outputs
save(mod.reduced, mod.full, gof.reduced, gof.full, file = "output/chemg_ergm.RData")
```

D.2.3 Comparing Fit across Models

```
# Assessing fit of cross-layer terms
dyadfit.observed <- summary(nw ~duplexdyad(type = letters[c(5 ,6, 7, 8)]))
dyadfit.reduced <- simulate(mod.reduced, nsim = 1000,
                            monitor = ~duplexdyad(type = letters[c(5 ,6, 7, 8)]),
                            statsonly = T, seed = 358946)[,c(23:26)]
dyadfit.full <- simulate(mod.full, nsim = 1000, statsonly = T, seed = 71905)[,c(21:24)]
```

```
# Output Table 2
sink(file = "output/chemg_fit.table.txt")
cbind(dyadfit.observed,
      colMeans(dyadfit.reduced), apply(dyadfit.reduced, 2, sd),
      colMeans(dyadfit.full), apply(dyadfit.full, 2, sd))
sink()

# Assess fit of different actor-dyad structures
# Function to count observed dyad census categories
dyad.census.count <- function(nw){
    mat <- as.matrix.network.adjacency(nw)
    nnodes <- nrow(mat)/2
    dyads <- combn(nnodes, 2)
    counts <- rep(NA, ncol(dyads))
    for(i in 1:length(counts)){
        dyad <- c(mat[dyads[1, i], dyads[2, i]],
                  mat[dyads[2, i], dyads[1, i]],
                  mat[dyads[1, i] + nnodes, dyads[2, i] + nnodes],
                  mat[dyads[2, i] + nnodes, dyads[1, i] + nnodes])
        if(sum(dyad) == 4){counts[i] <- 9}
        if(sum(dyad) == 0){counts[i] <- 10}
        if(sum(dyad) == 1){counts[i] <- ifelse(sum(dyad[1:2]) == 1, 1, 2)}
        if(sum(dyad) == 3){counts[i] <- ifelse(sum(dyad[1:2]) == 1, 8, 7)}
        if(sum(dyad) == 2){
            if(sum(dyad[1:2]) == 1){
                counts[i] <- ifelse(dyad[1] == dyad[3], 5, 6)
            }else{
                counts[i] <- ifelse(sum(dyad[1:2]) == 0, 4, 3)
            }
        }
    }
    tabulate(counts)
}

# Function to count dyad census for nsim simulated networks from supplied model
chemg.fit <- function(model, nsim = 1000, seed){
    nw.list<-simulate(model, verbose = F, nsim = nsim, seed = seed)
    t(sapply(nw.list,dyad.census.count))
}

# Running counting functions
observed.census <- dyad.census.count(nw)
mod.full.census <- chemg.fit(mod.full, nsim = 1000, seed = 2558821)
```

```
mod.reduced.census <- chemg.fit(mod.reduced, nsim = 1000, seed = 347938)

# Specifying colors for plotting
orng.c <- "#fcae91FF"; purp.c <- "#7828a0FF"; gren.c <- "#46aa96FF"
lgry.c <- "#dcdcdcFF"; mgry.c <- "#8c8c8cFF"; dgry.c <- "#333333FF"

# Plotting fit results (Figure 6)
pdf("output/chem_fitdensity.pdf", height = 10.5, width = 20)
par(oma = c(0, 0, 0, 0), omi = c(0.5, 0, 0, 0), mfrow = c(2, 5), mar = c(6, 1, 4, 1))
for(i in 1:10){
    den.full <- density(mod.full.census[, i], bw = "nrd")
    den.reduced <- density(mod.reduced.census[, i], bw = "nrd")
    den.full$y[1] <- 0
    den.reduced$y[1] <- 0
    plot.new()
    plot.window(xlim = range(c(den.full$x, den.reduced$x)),
                ylim = c(0, max(c(den.full$y, den.reduced$y))))
    abline(h = 0)
    abline(v = observed.census[i])
    polygon(den.reduced, col = adjustcolor(orng.c, alpha.f = 0.3),
            border = dgry.c, lty = 1, lwd = 2)
    polygon(den.full, col = adjustcolor(purp.c, alpha.f = 0.3),
            border = dgry.c, lty = 5, lwd = 2)

    box(); axis(1, cex.axis = 2.5, tick = F, line = 1)
    title(main = c(LETTERS[1:9], 0)[i], cex.main = 4)
}
par(fig=c(0, 1, 0, 1), oma=c(0, 0, 0, 0), mar=c(0, 0, 0, 0), new = T)
plot(0, 0, type = "n", bty = "n", xaxt = "n", yaxt = "n")
legend("bottomright", xpd = T, horiz = T, bty = "n", pch = 22,
       pt.bg = adjustcolor(c(orng.c, purp.c), alpha.f = 0.3), pt.lwd = 0.5,
       lty = c(1, 5), lwd = 2, pt.cex = 8, seg.len = 1,
       legend = c("Layer Independence", "Cross-Layer Dependence"),
       cex = 3.5)
dev.off()
```

D.3 Global Conflict Application

The global conflict application focuses on the tendency for conflicts to cluster. It is loosely based on Gleditsch, Salehyan and Schultz (2008). Data obtained from their replication files,[3]

[3]https://doi.org/10.1177/0022002707313305

Correlates of War, and the Uppsala Conflict Data Program. The code in D.3 takes approximately 45 hours to run using Amazon Web Service's C5 instances.

D.3.1 Data Preparation

```
rm(list = ls())
set.seed(20842)

# Organizing into multilayer structure.
# Interstate conflict on layer 1, nonstate conflict on layer 2,
# and civil conflict as interlayer ties.
nw <- cbind(rbind(mid, t(intraconf)), rbind(intraconf, nsconf))

# Into network class
nw <- network(nw, directed = F)
nw%v%"layer.mem" <- c(rep(1, 189), rep(2, 544))
nw%v%"dem" <- nodeatts$dem
nw%v%"cinc" <- nodeatts$cinc
nw%v%"trans" <- nodeatts$trans
nw%v%"interstate" <- nodeatts$interstate
nw%v%"intrastate" <- nodeatts$intrastate
nw%v%"nonstate" <- nodeatts$nonstate

# Constraining the few dyads that never coexist
absent <- matrix(0, ncol = 733, nrow = 733)
absent[1:189, 1:189] <- (1 - intoverlap)
diag(absent) <- 1
absent <- network(absent, directed = F)
```

D.3.2 Fitting ERGMs

```
# Layer Independence Model
mod.reduced <- ergm(nw
                    ~edgecov_layer("edges", layer = 1)
                    +degree_layer(0, layer = 1)
                    +altkstar.fixed_layer(layer = 1)
                    +nodefactor_layer("dem", layer = 1)
                    +nodefactor_layer("intrastate", layer = 1)
                    +nodecov_layer("cinc", layer = 1)
                    +edgecov_layer(demdem, layer = 1)
                    +edgecov_layer(contig, layer = 1)
```

```
                    +edgecov_layer(polrel, layer = 1)
                    +edgecov_layer(caprat, layer = 1)
                    +edgecov_layer("edges", layer = c(1, 2))
                    +altkstar.fixed_layer(layer = c(1,2))
                    +nodefactor_layer("dem", layer = c(1, 2))
                    +nodefactor_layer("trans", layer = c(1, 2))
                    +nodefactor_layer("interstate", layer = c(1, 2))
                    +nodefactor_layer("nonstate", layer = c(1, 2))
                    +edgecov_layer("edges", layer = 2)
                    +degree_layer(0, layer = 2)
                    +kstar_layer(2, layer = 2)
                    +altkstar.fixed_layer(layer = 2)
                    +nodefactor_layer("intrastate", layer = 2),
                    # Settings
                    eval.loglik = T, check.degeneracy = T, verbose = T,
                    control = control.ergm(seed = 80672,
                                           MCMC.burnin = 100000,
                                           MCMC.samplesize = 20000,
                                           MCMC.interval = 10000,
                                           parallel = 4),
                    constraints = ~fixedas(absent = absent))

# Model summary output (Table 3)
sink(file = "output/globalconflict_mod.reduced.summary.txt")
summary(mod.reduced)
sink()

# Model fit output
gof.reduced <- gof(mod.reduced,
                   control = control.gof.ergm(nsim = 1000, seed = 1208711))
pdf("output/globalconflict_mod.reduced.gof.pdf", height = 8, width = 8)
plot(gof.reduced)
dev.off()

pdf("output/globalconflict_mod.reduced.mcmc.pdf", height = 8, width = 8)
mcmc.diagnostics(mod.reduced)
dev.off()

# Cross-layer Dependence Model
mod.full <- ergm(nw
                 ~edgecov_layer("edges", layer = 1)
                 +degree_layer(0, layer = 1)
                 +altkstar.fixed_layer(layer = 1)
```

```
                  +nodefactor_layer("dem", layer = 1)
                  +nodecov_layer("cinc", layer = 1)
                  +edgecov_layer(demdem, layer = 1)
                  +edgecov_layer(contig, layer = 1)
                  +edgecov_layer(polrel, layer = 1)
                  +edgecov_layer(caprat, layer = 1)
                  +edgecov_layer("edges", layer = c(1, 2))
                  +altkstar.fixed_layer(layer = c(1,2))
                  +nodefactor_layer("dem", layer = c(1, 2))
                  +nodefactor_layer("trans", layer = c(1, 2))
                  +edgecov_layer("edges", layer = 2)
                  +degree_layer(0, layer = 2)
                  +kstar_layer(2, layer = 2)
                  +altkstar.fixed_layer(layer = 2)
                  +altkstar.fixed_crosslayer(lambda = 1,
                                             layers = list(1,c(1,2)))
                  +altkstar.fixed_crosslayer(lambda = 1,
                                             layers = list(c(1,2),2))
                  +threetrail_crosslayer(layers = list(1,c(1,2),2),
                                         incident = 1),
                  # Settings
                  eval.loglik = T, check.degeneracy = T, verbose = T,
                  control = control.ergm(seed = 30748,
                                         MCMC.burnin = 100000,
                                         MCMC.samplesize = 20000,
                                         MCMC.interval = 10000,
                                         parallel = 4),
                  constraints = ~fixedas(absent = absent))

# Model summary output (Table 3)
sink(file = "output/globalconflict_mod.full.summary.txt")
summary(mod.full)
sink()

# Model fit output
gof.full <- gof(mod.full, control = control.gof.ergm(nsim = 1000, seed = 707427))
pdf("output/globalconflict_mod.full.gof.pdf", height = 8, width = 8)
plot(gof.full)
dev.off()

pdf("output/globalconflict_mod.full.mcmc.pdf", height = 8, width = 8)
mcmc.diagnostics(mod.full)
dev.off()
```

```
# Saving results
save(mod.reduced, mod.full, gof.reduced, gof.full,
     file = "output/globalconflict_ergm.RData")
```

D.3.3 Comparing Fit across Models

```
# Assessing model fit to different domestic clustering scenarios
# Function to count interstate conflict ties according to domestic clustering
conflict.environment.count <- function(nw){
# Internal functions
# Determining the domestic conflict environments of a particular interstate conflict dyad
    strat.environment <- function(interstate, intrastate, nonstate){
        # Is the node in civil conflict?
        ns.conf <- function(c.node, intrastate, nonstate){
            ifelse(sum(nonstate[which(intrastate[c.node,] == 1),]) > 1, 1, 0)}
# Are the civil conflict partners of two countries the same?
        ns.same <- function(interstate, intrastate){
            ifelse(sum(which(intrastate[interstate[1],] == 1)
                   %in% which(intrastate[interstate[2],] == 1)) > 0, 1, 0)}
# Are the civil conflict partners of two countries
# in conflict with each other?
        ns.joint <- function(interstate, intrastate, nonstate){
            ifelse(sum(which(nonstate[which(intrastate[interstate[1],] == 1),,
                                      drop = F] == 1, arr.ind = T)[,2] %in%
                       which(nonstate[which(intrastate[interstate[2],] == 1),,
                                      drop = F] == 1, arr.ind = T)[,2]) > 0, 1, 0)}
# Are the countries in civil conflict?
        civ1 <- ifelse(sum(intrastate[interstate[1],]) > 0, 1, 0)
        civ2 <- ifelse(sum(intrastate[interstate[2],]) > 0, 1, 0)
# If both are not in civil conflict:
        if(civ1 + civ2 == 0){e.type <- 0}else{
# If only one is in civil conflict:
            if(civ1 + civ2 == 1){
                if(ns.conf(interstate[which(c(civ1, civ2) == 1)],
                           intrastate, nonstate) == 0){
# If non-state side of civil conflict is not in a nonstate conflict
                    e.type <- 1
                }else{
# If non-state side of civil conflict is in a nonstate conflict
                    e.type <- 2
                }
```

```
            }else{
# If both are in civil conflict:
                if(ns.same(interstate, intrastate) == 1){
# If countries share civil conflict partners:
                    e.type <- 6
                }else{
# If countries do not share civil conflict partners:
                    if(ns.conf(interstate[1], intrastate, nonstate)
                       + ns.conf(interstate[2], intrastate, nonstate) == 0){
# If civil conflict partners are not in nonstate conflicts:
                        e.type <- 3
                    }else{
# If civil conflict partners are in nonstate conflicts:
                        if(ns.joint(interstate, intrastate, nonstate) == 1){
# If civil conflict partners are fighting with each other:
                            e.type <- 5
                        }else{
# If civil conflict partners are not fighting each other:
                            e.type <- 4
                        }
                    }
                }
            }
        }
        return(e.type)
    }
    mat <- as.matrix.network.adjacency(nw)
    layer.mem <- get.node.attr(nw, "layer.mem")
    n.s <- sum(layer.mem == 1)
    n.ns <- sum(layer.mem == 2)
    dyads <- unique(t(apply(which(mat[1:n.s, 1:n.s] == 1, arr.ind = T), 1, sort)))
    counts <- apply(dyads, 1, strat.environment,
                    intrastate = mat[1:n.s, (n.s + 1):(n.s + n.ns)],
                    nonstate = mat[(n.s + 1):(n.s + n.ns), (n.s + 1):(n.s + n.ns)])
    tabulate(counts+1, 7)
}

# Function to determine interstate clustering distribution
# for nsim simulated networks from supplied model
globalconflict.fit <- function(model, nsim = 1000, seed){
    nw.list <- simulate(model, verbose = F, nsim = nsim, seed = seed)
    t(sapply(nw.list, conflict.environment.count))
}
```

```
# Running counting functions
observed.environments <- conflict.environment.count(nw)
mod.full.environments <- globalconflict.fit(mod.full, nsim = 1000, seed = 20843)
mod.reduced.environments <- globalconflict.fit(mod.reduced, nsim = 1000, seed = 606711)

# Specifying colors for plotting
orng.c <- "#fcae91FF"; purp.c <- "#7828a0FF"; gren.c <- "#46aa96FF"
lgry.c <- "#dcdcdcFF"; mgry.c <- "#8c8c8cFF"; dgry.c <- "#333333FF"

# Plotting fit results (Figure C2)
pdf("output/cf_fitdensity.pdf", height = 10, width = 16)
par(oma = c(0, 0, 0, 0), omi = c(0, 0, 0, 0), mfrow = c(2, 4), mar = c(6, 1, 4, 1))
for(i in c(1:7)){
    den.full <- density(mod.full.environments[, i], bw = ifelse(i > 5, "nrd0", "nrd"))
    den.reduced <- density(mod.reduced.environments[, i],
                           bw = ifelse(i > 5, "nrd0", "nrd"))
    den.full$y[1] <- 0
    den.reduced$y[1] <- 0
    plot.new()
    plot.window(xlim = range(c(den.full$x, den.reduced$x, 4)),
                ylim = c(0, max(c(den.full$y, den.reduced$y))))
    abline(h=0)
    abline(v = observed.environments[i])
    polygon(den.reduced, col = adjustcolor(orng.c, alpha.f = 0.3),
            border = dgry.c, lty = 1, lwd = 2)
    polygon(den.full, col = adjustcolor(purp.c, alpha.f = 0.3),
            border = dgry.c, lty = 5, lwd = 2)
    box();axis(1,cex.axis=2.5,tick=F,line=1)
    title(main=c(LETTERS[1:7])[i],cex.main=4)
}
plot.new()
plot.window(xlim=c(0,10),ylim=c(0,10))

legend("topleft", xpd = T, bty = "n", pch = 22,
       pt.bg = adjustcolor(orng.c, alpha.f = 0.3), pt.lwd = 0.5,
       lty = 1, lwd = 2, pt.cex = 8, seg.len = 1, col = dgry.c,
       legend = "Layer\nIndependence", cex = 3.5)
legend("left", xpd = T, bty = "n", pch = 22,
       pt.bg = adjustcolor(purp.c, alpha.f = 0.3), pt.lwd = 0.5,
       lty = 5, lwd = 2, pt.cex = 8, seg.len = 1, col = dgry.c,
       legend = "Cross-Layer\nDependence", cex = 3.5)
dev.off()
```

References

Allansson, Marie, Erik Melander and Lotta Themnér. 2017. "Organized violence, 1989–2015." *Journal of Peace Research* 54(4):727–742.

Braithwaite, Alex. 2005. "Location, location, location... identifying hot spots of international conflict." *International Interactions* 31(3):251–273.

Buhaug, Halvard and Kristian Skrede Gleditsch. 2008. "Contagion or confusion? Why conflicts cluster in space." *International Studies Quarterly* 52(2):215–233.

Cranmer, Skyler J and Bruce A Desmarais. 2011. "Inferential Network Analysis with Exponential Random Graph Models." *Political Analysis* 19(1):66–86.

Cunningham, David E, Kristian Skrede Gleditsch and Idean Salehyan. 2013. "Non-state actors in civil wars: A new dataset." *Conflict Management and Peace Science* 30(5):516–531.

Fjelde, Hanne and Desirée Nilsson. 2012. "Rebels against rebels: Explaining violence between rebel groups." *Journal of Conflict Resolution* 56(4):604–628.

Gibler, Douglas M and Alex Braithwaite. 2013. "Dangerous neighbours, regional territorial conflict and the democratic peace." *British Journal of Political Science* 43(4):877–887.

Gleditsch, Kristian Skrede. 2007. "Transnational dimensions of civil war." *Journal of Peace Research* 44(3):293–309.

Gleditsch, Kristian Skrede, Idean Salehyan and Kenneth Schultz. 2008. "Fighting at home, fighting abroad: How civil wars lead to international disputes." *Journal of Conflict Resolution* 52(4):479–506.

Koskinen, Johan and Galina Daraganova. 2013. Exponential Random Graph Model Fundamentals. In *Exponential Random Graph Models for Social Networks: Theory, Methods, and Applications*, ed. Dean Lusher, Johan Koskinen and Garry Robins. Cambridge University Press pp. 49–76.

Leifeld, Philip and Volker Schneider. 2012. "Information exchange in policy networks." *American Journal of Political Science* 56(3):731–744.

Maoz, Zeev. 2005. "Dyadic MID Dataset (DYMID) Dataset (version 2.0)." *Computer File. http://psfaculty. ucdavis. edu/zmaoz/dyadmid. html* .

Robins, Garry, Tom Snijders, Peng Wang, Mark Handcock and Philippa Pattison. 2007. "Recent developments in exponential random graph (p*) models for social networks." *Social networks* 29(2):192–215.

Snijders, Tom AB, Philippa E Pattison, Garry L Robins and Mark S Handcock. 2006. "New Specifications for Exponential Random Graph Models." *Sociological Methodology* 36(1):99–153.

Wang, Peng, Garry Robins and Petr Matous. 2016. Multilevel Network Analysis Using ERGM and Its Extension. In *Multilevel Network Analysis for the Social Sciences*. Springer pp. 125–143.

Wang, Peng, Garry Robins, Philippa Pattison and Emmanuel Lazega. 2013. "Exponential Random Graph Models for Multilevel Networks." *Social Networks* 35(1):96–115.