R Package iglm: Regression under Interference in Connected Populations

Cornelius Fritz
School of Computer Science and Statistics
Trinity College Dublin
Ireland

Michael Schweinberger
Department of Statistics
The Pennsylvania State University
USA

Abstract

We introduce R package **iglm**, which implements a comprehensive framework for studying relationships among predictors and outcomes under interference. The implemented regression framework facilitates the study of spillover and other phenomena in connected populations and has important advantages over existing packages, among them scalability and provable theoretical guarantees. On the computational side, the regression framework relies on scalable methods that can be applied to small and large data sets, by solving a convex optimization program based on pseudo-likelihoods using Minorization-Maximization and Quasi-Newton algorithms. On the statistical side, the regression framework comes with provable theoretical guarantees. To increase the versatility of **iglm**, users can add custom-built model terms. We showcase **iglm** using two data sets, including hate speech on the social media platform X and communications among students.

Keywords: Dependent Data, Generalized Linear Model, Minorization-Maximization, Quasi-Newton, Social Network, Spillover.

1. Regression under Interference

Linear and generalized linear models are widely used in data science, helping study relationships among predictors X and outcomes Y and make model-based predictions. These regression models, implemented in R functions `lm()` and `glm()` (R Core Team 2026) and other statistical software packages, assume that the attributes (X_i, Y_i) of any unit i are unaffected by the attributes (X_j, Y_j) of all other units j in the population of interest. Such independence assumptions may be violated in connected populations due to spillover and other phenomena that induce dependence among attributes of connected population members. For example, if teenager i is exposed to advertisements of designer clothes on social media ($X_i = 1$), then i may purchase the advertised designer clothes ($Y_i = 1$). The effect of X_i on Y_i may spill

over to other teenagers: e.g., teenager i's friend j may observe i wearing the purchased designer clothes ($Y_i = 1$) and may decide to purchase them as well ($Y_j = 1$). As a result, the outcomes Y_i and Y_j of teenagers i and j are dependent conditional on the exposures X_i and X_j of i and j to advertisements, provided that i and j are connected ($Z_{i,j} = 1$). Studying relationships among attributes (X_i, Y_i) in connected populations requires a comprehensive regression framework for dependent attributes (X_i, Y_i) and connections $Z_{i,j}$, including fixed designs (with X_i or $Z_{i,j}$ fixed) and random designs (with X_i and $Z_{i,j}$ random).

1.1. R Package iglm

R package **iglm** (Fritz and Schweinberger 2025) provides a comprehensive regression framework for dependent attributes (X_i, Y_i) and connections $Z_{i,j}$ with fixed and random designs, building on Fritz, Schweinberger, Bhadra, and Hunter (2026). The comprehensive regression framework implemented in **iglm** comes with multiple benefits, first and foremost scalability and provable theoretical guarantees. We describe these benefits in Section 1.3, but we first compare **iglm** to other packages in Section 1.2.

1.2. Comparison with Other Packages

While no comprehensive framework for regression in connected populations exists, there are at least three joint probability models for dependent outcomes Y_i and connections $Z_{i,j}$, in addition to numerous conditional models for dependent outcomes Y_i given connections $Z_{i,j}$ and conditional models for dependent connections $Z_{i,j}$ given predictors X_i. Examples of conditional models for dependent outcomes Y_i given connections $Z_{i,j}$ include autologistic actor attribute models (ALAAMs), implemented in the software package **PNet** (Wang, Robins, Pattison, and Koskinen 2009; Koskinen and Daraganova 2022). Conditional models for dependent connections $Z_{i,j}$ given predictors X_i include additive and multiplicative effects models (Hoff 2021) implemented in R package **amen** (Hoff, Fosdick, and Volfovsky 2024) and exponential random graph models (ERGMs) implemented in R packages **ergm** and **Bergm** (Krivitsky, Hunter, Morris, and Klumb 2023; Caimo and Friel 2014). Among the joint probability models for dependent outcomes Y_i and connections $Z_{i,j}$ are continuous-time Markov processes for discrete and continuous outcomes $Y_{i,t}$ observed at two or more time points $t \in \{1, \ldots, T\}$ ($T \geq 2$) (Snijders, Steglich, and Schweinberger 2007; Niezink and Snijders 2017), which are implemented in R package **RSiena** (Snijders, Ripley, Boitmanis, Steglich, Niezink, Amati, and Schoenenberger 2025); exponential random network models (ERNMs) (Fellows and Handcock 2012; Wang, Fellows, and Handcock 2024), which can be viewed as ERGMs with categorical outcomes Y_i and are implemented in R package **ernm** (Fellows and Clark 2025); and multivariate normal models for continuous outcomes Y_i (Fosdick and Hoff 2015), which are not implemented in publicly available software. While all of these approaches have useful applications, their objectives, models, and methods differ from those of R package **iglm**. First, none of the mentioned approaches provides a comprehensive regression framework for binary, count-, and real-valued attributes (X_i, Y_i) and connections $Z_{i,j}$. Second, the mentioned approaches rely on Markov chain Monte Carlo simulations for estimating models, which are time-consuming and make the statistical analysis of large data sets challenging. Third, the proposed estimators do not come with provable theoretical guarantees based on a single observation of dependent attributes (X_i, Y_i) and connections $Z_{i,j}$: e.g., consistency results and rates of convergence for estimators are unavailable. Last, but not least,

the non-Bayesian approaches quantify the uncertainty about estimators based on standard asymptotics, using the inverse Fisher or Godambe information matrix. Having said that, standard asymptotics based on models for independent attributes (X_i, Y_i) with a fixed number of weights $p < \infty$ may not be applicable to non-standard models for dependent attributes (X_i, Y_i) and connections $Z_{i,j}$ with an increasing number of weights $p \to \infty$ (e.g., when unit-dependent weights are used to capture unobserved heterogeneity in terms of connectivity). As a consequence, conclusions based on standard asymptotics can be misleading.

1.3. Advantages of R Package iglm

R package **iglm** implements a comprehensive regression framework for dependent attributes (X_i, Y_i) and connections $Z_{i,j}$ with important advantages over existing packages, first and foremost scalability and provable theoretical guarantees:

- **Comprehensive framework for attributes (X_i, Y_i) and connections $Z_{i,j}$:** R package **iglm** provides data scientists with a comprehensive regression framework for binary, count-, and real-valued attributes (X_i, Y_i) and connections $Z_{i,j}$. These regression models allow attributes (X_i, Y_i) and connections $Z_{i,j}$ to be dependent, enabling data scientists to study spillover and other phenomena in connected populations.

- **Interpretable framework:** The regression framework implemented in R package **iglm** can be viewed as an extension of generalized linear models (GLMs) from independent attributes (X_i, Y_i) to dependent attributes (X_i, Y_i) and connections $Z_{i,j}$. As a consequence, users can interpret results along the lines of GLMs, including logistic regression, Poisson regression, and linear regression models.

- **Local dependence in small and large populations:** R package **iglm** allows users to leverage additional structure in the form of neighborhoods, which helps construct models with local dependence among attributes (X_i, Y_i) and connections $Z_{i,j}$ in overlapping neighborhoods. Local dependence respects the local nature of small and large populations and helps control dependence, facilitating theoretical guarantees. R package **iglm** therefore provides an extensive list of model terms with local dependence among attributes (X_i, Y_i) and connections $Z_{i,j}$.

- **Extendability:** Users can contribute custom-built model terms to R package **iglm**, making it a versatile platform for regression under interference in connected populations.

- **Scalability:** R package **iglm** can estimate models from small and large data sets by solving a convex optimization problem based on pseudo-likelihoods using Minorization-Maximization and Quasi-Newton methods, without requiring time-consuming Markov chain Monte Carlo methods. These methods enable applications with up to 10,000 units and large-scale simulation studies, as demonstrated by Fritz *et al.* (2026).

- **Uncertainty quantification:** R package **iglm** quantifies the uncertainty about maximum pseudo-likelihood estimators based on their exact covariance matrix, without relying on asymptotic normality or other standard asymptotics that may not be applicable to non-standard models for dependent attributes (X_i, Y_i) and connections $Z_{i,j}$.

- **Provable theoretical guarantees:** The regression models and methods implemented in R package **iglm** are supported by provable theoretical guarantees: e.g., Corollary 1

of Fritz *et al.* (2026) shows that the error of maximum pseudo-likelihood estimators of $p = O(N)$ weights decays at rate $\sqrt{\log N/N}$ under models with local dependence among attributes (X_i, Y_i) and connections $Z_{i,j}$ in overlapping subpopulations. Simulation results support these theoretical results.

1.4. Architecture of R Package iglm

R package **iglm** is a free, publicly available, and platform-independent R package that can be downloaded from the website `https://cran.r-project.org` of the Comprehensive R Archive Network (CRAN) (R Core Team 2026). It can be installed and loaded in R as follows:

```
R> install.packages("iglm")
R> library(iglm)
```

R package **iglm** leverages `R6` classes (Chang 2025) for three reasons:

- **Model and data classes:** The `R6` classes `iglm` and `iglm.data` bundle all `R` functions belonging to a class in so-called methods. For example, upon constructing an `iglm` model called `m`, one can use `R` functions `m$estimate()`, `m$simulate()`, and `m$assess()` to estimate, simulate, and assess model `m`, respectively.

- **Design consistency:** The design of the `R6` class `iglm` places a strong emphasis on consistency between nested classes. For example, all results are saved in the nested `R6` class `results`, which is consistent with the main object `iglm`: e.g., the command `m$results$plot(trace = TRUE)` provides trace plots of all estimates and if the model is updated, estimates and model-based predictions under different models will be deleted.

- **Natural data sharing:** R package **iglm** allows users to share data across multiple stages of data analysis. For example, simulated data are used in two stages of data analysis—to quantify the uncertainty about estimators and to assess models based on simulated data—and R package **iglm** shares samples across these two stages of analysis.

In addition to its dependency on R package **R6** classes (Chang 2025), **iglm** depends on **coda** (Plummer, Best, Cowles, and Vines 2006) to save statistics; **igraph** (Csárdi, Nepusz, Traag, Horvát, Zanini, Noom, and Müller 2025) to plot data; and **Rcpp** (Eddelbuettel 2013) and **RcppArmadillo** (Eddelbuettel and Sanderson 2014) to speed up computing by outsourcing time-consuming tasks to C++. All results are based on R package **iglm** version 1.2.4 and R version 4.5.2, using a MacBook Pro with macOS Tahoe 26.4.1. Since pseudo-random number generators can vary across operating systems and R versions, running the replication script on other operating systems or R versions can result in small differences.

1.5. Structure

We describe the data required by R package **iglm** in Section 2 and review models, methods, and implementation in Section 3, using data on hate speech on the social media platform X with binary outcomes $Y_i \in \{0, 1\}$ and connections $Z_{i,j} \in \{0, 1\}$ as a running example. Advanced topics are discussed in Section 4, including model comparison and custom-built

model terms. An additional application to real-valued outcomes $Y_i \in \mathbb{R}$ and connections $Z_{i,j} \in \{0, 1\}$ is presented in Section 5.

2. Data

R package **iglm** considers the following data based on $N \geq 2$ units, stored in data object `iglm.data`:

- **Predictors** $\boldsymbol{X} = (X_i)$: A vector `x_attribute` of length N, storing unit-level predictors. The data type is specified by the `type_x` argument, which can be binary ($X_i \in \{0, 1\}$: `type_x = "binomial"`), count-valued ($X_i \in \{0, 1, \ldots\}$: `type_x = "poisson"`), or real-valued ($X_i \in \mathbb{R}$: `type_x = "normal"`). The predictors X_i are random: e.g., X_i may be the treatment assignment of unit i in a randomized experiment. At the time of writing, R package **iglm** supports a single random predictor X_i, but an arbitrary number of non-random predictors can be added to the local R environment, including unit-dependent predictors (e.g., the demographic background of units i) or dyad-dependent predictors (e.g., indicators of whether pairs of units i and j have a shared demographic background). Since non-random predictors are viewed as exogenous, R package **iglm** does not store them in data object `iglm.data`.

- **Outcomes** $\mathbf{Y} = (Y_i)$: A vector `y_attribute` of length N, storing unit-level outcomes. The data type is specified by the `type_y` argument, which can be binary ($Y_i \in \{0, 1\}$: `type_y = "binomial"`), count-valued ($Y_i \in \{0, 1, \ldots\}$: `type_y = "poisson"`), or real-valued ($Y_i \in \mathbb{R}$: `type_y = "normal"`).

- **Connections** $\boldsymbol{Z} = (Z_{i,j})$: A connection between units i and j is indicated by $Z_{i,j} \in \{0, 1\}$, where $Z_{i,j} \coloneqq 1$ indicates that units i and j are connected and $Z_{i,j} \coloneqq 0$ otherwise. The connections $Z_{i,j}$ can be directed or undirected, in which case $Z_{i,j} = Z_{i,j}$ for all pairs of units i and j. Self-connections are excluded by setting $Z_{i,i} \coloneqq 0$ for all units i. The directed or undirected connections can be represented by an $N \times N$-adjacency matrix $\boldsymbol{Z} \coloneqq (Z_{i,j}) \in \{0, 1\}^{N \times N}$ or an $M \times 2$-edge matrix $\boldsymbol{E} \coloneqq (e_{m,k}) \in \{1, \ldots, N\}^{M \times 2}$, where $M \geq 1$ is the number of observed edges and $E_{m,1} = i$ and $E_{m,2} = j$ indicates that the mth observed edge connects units i and j. In the case of undirected connections, the $M \times 2$-edge matrix lists all pairs of units i and j with $i < j$ such that $Z_{i,j} = 1$, and the $N \times N$-adjacency matrix $\boldsymbol{Z}$ is symmetric. If N is large and the network is sparse, edge lists require less memory than adjacency matrices and are hence preferable.

- **Neighborhoods** $\mathcal{N}_1, \ldots, \mathcal{N}_N$ **(optional but recommended):** If additional structure in the form of neighborhoods $\mathcal{N}_1, \ldots, \mathcal{N}_N$ of units $1, \ldots, N$ is available, it can be imported by specifying the option `neighborhood`; note that these neighborhoods are viewed as exogenous and should not based on the network $\boldsymbol{Z}$. The neighborhood $\mathcal{N}_i$ of unit i consists of the subset of all other units that can affect the outcome Y_i and connections $Z_{i,j}$ of unit i. If no neighborhood is provided, it is assumed that the neighborhood $\mathcal{N}_i$ of unit i consists of all other units, which implies that the attributes (X_i, Y_i) and connections $Z_{i,j}$ of unit i can be affected by all other units. The neighborhood structure can be provided in the same form as the connections, i.e., either in the form of an $N \times N$-adjacency matrix or as an $L \times 2$-edge matrix specifying which pairs of units i and j are neighbors, where L denotes the total number of pairs of neighbors.

Throughout, $\mathbb{I}(\cdot)$ is an indicator function, which is 1 if its argument is true and is 0 otherwise, and $c_{i,j} \coloneqq \mathbb{I}(\mathcal{N}_i \cap \mathcal{N}_j \neq \emptyset)$ is an indicator of whether the neighborhoods $\mathcal{N}_i$ and $\mathcal{N}_j$ of units i and j overlap. We assume that the neighborhoods are known and discuss extensions to unknown neighborhoods in Section 6.

Example: Hate speech on X As a running example, we use data collected by Kim, Nakka, Gopal, Desmarais, Mancinelli, Harden, Ko, and Boehmke (2022). The data concern hate speech by $N = 495$ U.S. state legislators on the social media platform X in the six months preceding the January 6, 2021 insurrection at the U.S. Capitol. We use a subset of the data, consisting of legislators in California (CA), Florida (FL), Illinois (IL), New York (NY), Pennsylvania (PA), and Texas (TX). The outcome $Y_i \in \{0, 1\}$ indicates whether posts of legislator i were identified as hate speech by Large Language Models (Fritz *et al.* 2026), where $Y_i \coloneqq 1$ if one or more posts by legislator i were classified as hate speech and $Y_i \coloneqq 0$ otherwise. The predictor $X_i \in \{0, 1\}$ is $X_i \coloneqq 1$ if legislator i is Republican and $X_i \coloneqq 0$ otherwise. Other predictors include gender ($v_{i,1} \coloneqq 1$ if i is female and $v_{i,1} \coloneqq 0$ otherwise), race ($v_{i,2} \coloneqq 1$ if i is white and $v_{i,2} \coloneqq 0$ otherwise), and state ($v_{i,3} \in \{\text{CA, FL, IL, NY, PA, TX}\}$). On the social media platform X, users can mention or repost posts by others. We set $Z_{i,j} \coloneqq 1$ if legislator i mentioned or reposted posts by legislator j between January 6, 2020 and January 6, 2021 and set $Z_{i,j} \coloneqq 0$ otherwise. The connections $Z_{i,j}$ are directed, so $Z_{i,j}$ may not equal $Z_{j,i}$.

If the vectors `x_attribute` and `y_attribute` and the matrices `z_network` and `neighborhood` are available in the local R environment, we can instantiate data object `iglm.data` as follows:

```
R> data.object <- iglm.data(x_attribute = x_attribute,
+                           type_x = "binomial",
+                           y_attribute = y_attribute,
+                           type_y = "binomial",
+                           z_network = z_network,
+                           directed = TRUE,
+                           neighborhood = neighborhood)
```

The type of connections $Z_{i,j}$ does not need to be specified, because R package **iglm** assumes that all connections are binary, i.e., $Z_{i,j} \in \{0, 1\}$. As mentioned, the non-random predictors $v_{i,1}$, $v_{i,2}$, and $v_{i,3}$ are viewed as exogenous and are not stored in `data.object`, but these predictors can be added to the local R environment and can be used in models.

Descriptive Statistics Printing `data.object` provides descriptive statistics, including the number of units ($N = 495$), an indicator of whether the network is directed, the number of connected pairs of units ($\sum_{(i,j)\in\mathcal{D}} Z_{i,j} = 9{,}218$), the total size of all neighborhoods ($\sum_{i\in\mathcal{P}} |\mathcal{N}_i| = 24{,}398$), and summaries of attributes (X_i, Y_i):

```
R> data.object
iglm.data object
  units                       : 495
  directed                    : TRUE
  edges (fixed = FALSE)       : 9218
  neighborhood edges          : 24398
```

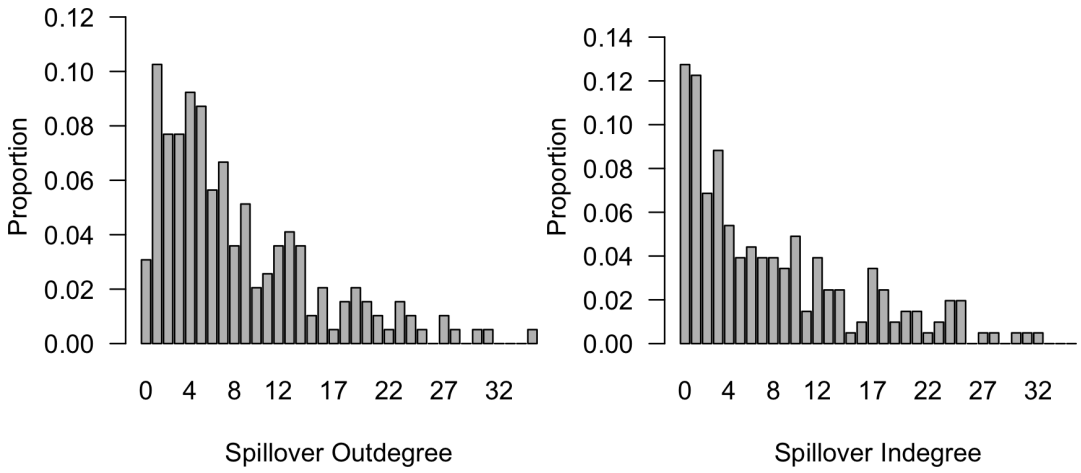

Figure 1: Hate speech on X: observed in- and outdegrees of U.S. state legislators in the subnetwork consisting of connections among pairs of legislators i and j with $c_{i,j}\, x_i\, y_j = 1$ or $c_{i,j}\, x_j\, y_i = 1$. Such connections represent the channels that enable spillover and therefore the in- and outdegrees in the described subnetwork are called spillover in- and outdegrees.

```
Attribute summaries
  x_attribute (fixed = FALSE) : binomial 1s=195, 0s=300, P(1)=0.394
  y_attribute                 : binomial 1s=204, 0s=291, P(1)=0.412
```

Users can explore the data in more depth using the following methods supplied by `iglm.data`:

- `x_distribution()` and `y_distribution()` compute and plot the empirical distributions of attributes X_i and Y_i.

- `degree_distribution(plot = TRUE)` computes and plots the degrees of units, i.e., the numbers of connections of units. If the network is directed, the function computes and plots both the distribution of indegrees (the numbers of incoming connections of units) and the distribution of outdegrees (the numbers of outgoing connections of units).

- `spillover_degree_distribution()` plots the spillover degree distribution. For binary predictors $X_i \in \{0, 1\}$ and outcomes $Y_i \in \{0, 1\}$, the spillover degree distribution is defined as the degree distribution in the subnetwork consisting of connections among pairs of units i and j with $c_{i,j}\, x_i\, y_j = 1$ or $c_{i,j}\, x_j\, y_i = 1$, which are the connections that enable spillover. The plots produced by `data.object$spillover_degree_distribution()` are shown in Figure 1. For non-binary attributes (X_i, Y_i), we compute the spillover degree distribution based on binarized attributes. A simple approach to binarizing non-binary attributes (X_i, Y_i) is based on $\widetilde{X}_i \coloneqq \mathbb{I}(X_i > \overline{X})$ and $\widetilde{Y}_i \coloneqq \mathbb{I}(Y_i > \overline{Y})$, where $\mathbb{I}(\cdot)$ is an indicator function that is 1 if its argument is true and is 0 otherwise, and $\overline{X}$ and $\overline{Y}$ are the arithmetic means of the respective attributes.

- `geodesic_distances_distribution()` computes and plots geodesic distances, i.e., the number of pairs of units with shortest path of length $1, 2, \ldots$

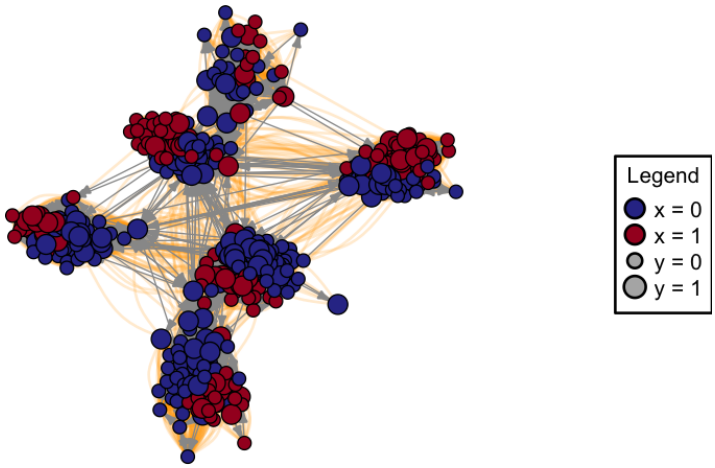

Figure 2: Hate speech on X: interactions among U.S. state legislators represented by lines between circles. The line between two legislators i and j is colored yellow if the neighborhoods $\mathcal{N}_i$ and $\mathcal{N}_j$ of i and j overlap and is otherwise colored gray. The colors and sizes of the circles represent the party affiliation X_i and the use of hate speech Y_i of legislators i.

- `dyadwise_shared_partner_distribution()` computes and plots the distribution of the numbers of pairwise shared partners, i.e., the number of pairs of units—either connected or unconnected—with $0, 1, \ldots$ shared partners.

- `edgewise_shared_partner_distribution()` computes and plots the distribution of the numbers of edgewise shared partners, i.e., the number of connected pairs of units with $0, 1, \ldots$ shared partners.

To visualize the data object, R package **iglm** relies on tools implemented in R package **igraph** (Csárdi *et al.* 2025):

```
R> set.seed(123)
R> data.object$plot(edge.width = 0.5, edge.arrow.size = 0.2,
+                   legend_size = 0.5, legend_size_n_levels = 2)
```

The resulting plot is shown in Figure 2.

3. Regression under Interference

We describe a comprehensive framework for regression under interference in connected populations and its implementation in R package **iglm**, including model specification, estimation, uncertainty quantification, simulation, assessment, and interpretation. A graphical representation of these topics can be found in Figure 3.

A comprehensive framework for regression under interference in connected populations can be based on probability measures $\{\mathbb{P}_{\boldsymbol{\theta}}, \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ dominated by a σ-finite measure $\nu_{\mathbf{x},\mathbf{y},\mathbf{z}}$ with

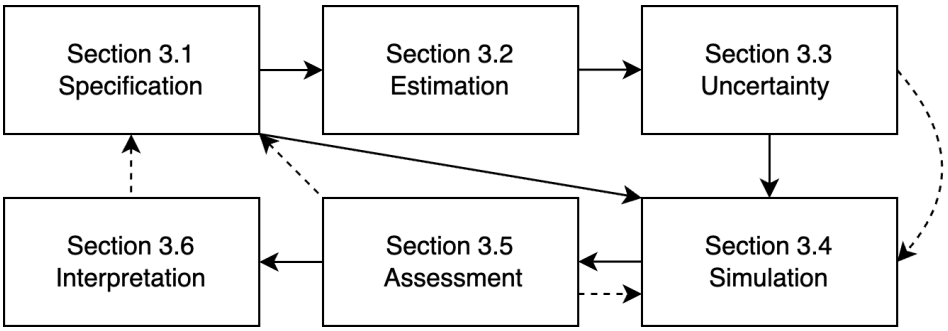

Figure 3: Solid arrows indicate the workflow of R package **iglm**, while dashed arrows indicate dependencies. For example, simulating data from the model helps quantify the uncertainty about estimators and assess the estimated model, while the interpretation and assessment of the model may suggest updating the specification. In addition, one may wish to simulate from a specified model, in which case estimation and uncertainty quantification can be bypassed.

densities $f_{\boldsymbol{\theta}} \coloneqq \mathrm{d}\,\mathbb{P}_{\boldsymbol{\theta}} / \mathrm{d}\,\nu_{\mathcal{X},\mathcal{Y},\mathcal{Z}}$ of the form

$$\begin{aligned} f_{\boldsymbol{\theta}}(\boldsymbol{x},\mathbf{y},\boldsymbol{z}) &= \frac{1}{c(\boldsymbol{\theta})}\left[\prod_{i\in\mathcal{P}} a_{\mathcal{X}}(x_i)\, a_{\mathcal{Y}}(y_i)\, \exp(\boldsymbol{\theta}_g^\top g_i(x_i,y_i))\right] \\ &\times \left[\prod_{(i,j)\in\mathcal{D}} a_{\mathcal{Z}}(z_{i,j})\, \exp(\boldsymbol{\theta}_h^\top h_{i,j}(x_i,x_j,y_i,y_j,\boldsymbol{z}))\right], \end{aligned}$$

where

- $\nu_{\mathcal{X},\mathcal{Y},\mathcal{Z}}$ is a σ-finite product measure of the form

$$\nu_{\mathcal{X},\mathcal{Y},\mathcal{Z}}(\boldsymbol{x},\mathbf{y},\boldsymbol{z}) \coloneqq \left[\prod_{i\in\mathcal{P}} \nu_{\mathcal{X}}(x_i)\, \nu_{\mathcal{Y}}(y_i)\right]\left[\prod_{(i,j)\in\mathcal{D}} \nu_{\mathcal{Z}}(z_{i,j})\right],$$

  where $\nu_{\mathcal{X}}$, $\nu_{\mathcal{Y}}$, and $\nu_{\mathcal{Z}}$ are σ-finite measures that depend on the sample spaces of X_i, Y_i, and $Z_{i,j}$ (e.g., counting measure when $Y_i \in \{0,1\}$ or Lebesgue measure when $Y_i \in \mathbb{R}$).

- $\mathcal{P} \coloneqq \{1,\ldots,N\}$ is the set of units and $\mathcal{D}$ is the set of pairs of units with possible connections:

$$\mathcal{D} \coloneqq \begin{cases} \{(i,j) : 1 \le i,\, j \le N,\, i \ne j\} & \text{if connections are directed} \\ \{(i,j) : 1 \le i < j \le N\} & \text{if connections are undirected.} \end{cases}$$

- $a_{\mathcal{X}}(\cdot)$, $a_{\mathcal{Y}}(\cdot)$, and $a_{\mathcal{Z}}(\cdot)$ are reference measures that determine the conditional distributions of X_i, Y_i, and $Z_{i,j}$, as demonstrated in Section 3.1.

- $g_i(\cdot)$ is a vector of unit-dependent functions that describe the relationship between predictors X_i and outcomes Y_i, with a corresponding vector of weights $\boldsymbol{\theta}_g$.

- $h_{i,j}(\cdot)$ is a vector of pairwise functions that specify how the attributes (X_i, Y_i) and (X_j, Y_j) of units i and j depend on each other, and how the connection $Z_{i,j}$ depends on other connections, with a corresponding vector of weights $\boldsymbol{\theta}_h$.
- $\boldsymbol{\theta} \coloneqq (\boldsymbol{\theta}_g, \boldsymbol{\theta}_h) \in \boldsymbol{\Theta}$ is the vector of all weights, where $\boldsymbol{\Theta} \coloneqq \{\boldsymbol{\theta} \in \mathbb{R}^p : c(\boldsymbol{\theta}) < \infty\}$ $(p \geq 1)$ and $c(\boldsymbol{\theta}) \coloneqq \int_{\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}} f_{\boldsymbol{\theta}}(\boldsymbol{x}, \mathbf{y}, \boldsymbol{z}) \,\mathrm{d}\, \nu_{\mathcal{X},\mathcal{Y},\mathcal{Z}}(\boldsymbol{x}, \mathbf{y}, \boldsymbol{z})$.

The regression framework described above can capture relationships among attributes (X_i, Y_i) of units i via $g_i(\cdot)$ and dependence among attributes (X_i, Y_i) and (X_j, Y_j) of connected pairs of units i and j $(Z_{i,j} = 1)$ via $h_{i,j}(\cdot)$. We demonstrate in Sections 3.1, 3.6, and 5 that these models can be viewed as a natural extension of GLMs from independent attributes (X_i, Y_i) to dependent attributes (X_i, Y_i) and connections $Z_{i,j}$. To ensure that regression models for dependent attributes (X_i, Y_i) and connections $Z_{i,j}$ are well-behaved in small and large populations and obtain theoretical guarantees for statistical procedures, we assume that $(\boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z})$ satisfies local dependence, in the sense that

$$h_{i,j}(x_i, x_j, y_i, y_j, \boldsymbol{z}) \quad = \quad h_{i,j}(z_{i,j}, z_{j,i}) \text{ for all } (i,j) \in \mathcal{D} \text{ such that } \mathcal{N}_i \cap \mathcal{N}_j = \emptyset.$$

In other words, if units i and j are not close in the sense that their neighborhoods $\mathcal{N}_i$ and $\mathcal{N}_j$ do not overlap, their attributes (X_i, Y_i) and (X_j, Y_j) are independent. The fact that $h_{i,j}(\cdot)$ is allowed to be a function of $z_{i,j}$ and $z_{j,i}$ helps capture reciprocity, i.e., the tendency to reciprocate connections in directed networks.

3.1. Specification

The class `iglm` supplies statistical tools for data analysis, including model specification, estimation, uncertainty quantification, simulation, and assessment. A graphical representation of class `iglm` can be found in Figure 4. The class `iglm` consists of three components:

- `formula` specifies the model;
- `control` specifies the estimation algorithm described in Section 3.2;
- `sampler` specifies the simulation algorithm described in Section 3.4.

In line with GLMs for independent attributes (X_i, Y_i) implemented in R function `glm()`, regression models for dependent attributes (X_i, Y_i) and connections $Z_{i,j}$ are specified by an object of type `formula`:

```
formula = data.object ~ term1 + term2 + ...
```

The left-hand side specifies the data in the form of `data.object`, while the right-hand side specifies the model using terms `term1` and `term2` separated by the symbol `+`. The dots `...` represent additional model terms. In contrast to GLMs for independent attributes (X_i, Y_i) implemented in R function `glm()`, the attributes (X_i, Y_i) and (X_j, Y_j) of units i and j with connections $Z_{i,j}$ can be dependent, and connections $Z_{i,j}$ can depend on other connections by including suitable model terms.

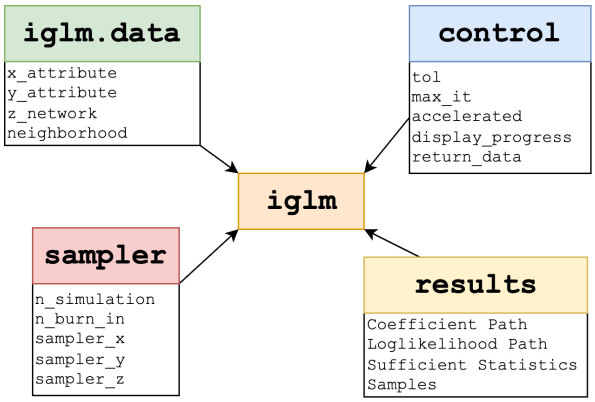

Figure 4: A graphical representation of class `iglm`, which encompasses objects `iglm.data`, `control`, `results`, and `sampler`. Additional options supplied by these objects are described in the help function of R package `iglm`: see `?iglm`.

Left-hand side of `formula`*: iglm.data*

The class `iglm.data` is described in Section 2. We focus here on the specification of the `type` of predictors X_i and outcomes Y_i, which determines the reference measures $a_{\mathcal{X}}(.)$ and $a_{\mathcal{Y}}(.)$. We first consider outcomes Y_i and write $\mathbf{Y}_{-i}$ to denote the set of outcomes $\mathbf{Y}$ excluding Y_i:

- `set_type_y("binomial")` sets $a_{\mathcal{Y}}(y_i) \coloneqq \mathbb{I}(y_i \in \{0,1\})$, which implies that the conditional distribution of $Y_i \mid (\boldsymbol{X}, \mathbf{Y}_{-i}, \boldsymbol{Z}) = (\boldsymbol{x}, \mathbf{y}_{-i}, \boldsymbol{z})$ is Bernoulli with mean $\exp(\eta_{\mathcal{Y},i})/(1+\exp(\eta_{\mathcal{Y},i}))$, where $\eta_{\mathcal{Y},i} \in \mathbb{R}$ is a linear predictor that depends on `formula`.
- `set_type_y("poisson")` sets $a_{\mathcal{Y}}(y_i) \coloneqq 1/(y_i!)\, \mathbb{I}(y_i \in \{0,1,\dots\})$, which implies that the conditional distribution of $Y_i \mid (\boldsymbol{X}, \mathbf{Y}_{-i}, \boldsymbol{Z}) = (\boldsymbol{x}, \mathbf{y}_{-i}, \boldsymbol{z})$ is Poisson with mean $\exp(\eta_{\mathcal{Y},i})$.
- `set_type_y("normal")` sets

$$a_{\mathcal{Y}}(y_i) \quad \coloneqq \quad \frac{1}{\sqrt{2\,\pi\,\psi_{\mathcal{Y}}}} \exp\left(-\frac{y_i^2}{2\,\psi_{\mathcal{Y}}}\right) \mathbb{I}(y_i \in \mathbb{R}),$$

  which implies that the conditional distribution of $Y_i \mid (\boldsymbol{X}, \mathbf{Y}_{-i}, \boldsymbol{Z}) = (\boldsymbol{x}, \mathbf{y}_{-i}, \boldsymbol{z})$ is Normal with mean $\eta_{\mathcal{Y},i}$ and variance $\psi_{\mathcal{Y}} > 0$.

In all three cases, the linear predictor $\eta_{\mathcal{Y},i}$ depends on the joint probability model of $(\boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z})$, which is specified by `formula`. We provide examples of linear predictors $\eta_{\mathcal{Y},i}$ when $Y_i \in \{0,1\}$ (Section 3.6) and $Y_i \in \mathbb{R}$ (Section 5).

As mentioned above, the conditional distributions of predictors X_i and outcomes Y_i can be represented by GLMs. The specific GLMs depend on the sample spaces and reference measures $a_{\mathcal{X}}(.)$ and $a_{\mathcal{Y}}(.)$ of X_i and Y_i. The conditional distributions of connections $Z_{i,j} \in \{0,1\}$ are Bernoulli, assuming that $a_{\mathcal{Z}}(z_{i,j}) \coloneqq 1$ if $z_{i,j} \in \{0,1\}$ and $a_{\mathcal{Z}}(z_{i,j}) \coloneqq 0$ otherwise, and can be represented by logistic regression models.

To provide additional flexibility, R package **iglm** allows users to fix $\boldsymbol{X}$ or $\boldsymbol{Z}$, using the following methods provided by `iglm.data`:

- `set_fix_x(TRUE)` fixes the predictors X_i of all units i at their observed values x_i, corresponding to a fixed covariate design.
- `set_fix_z(TRUE)` fixes all connections $Z_{i,j}$ among all pairs of units i and j at their observed values $z_{i,j}$, corresponding to a fixed network design.
- `set_fix_z_alocal(TRUE)` fixes connections $Z_{i,j}$ among all pairs of units i and j without overlapping neighborhoods $\mathcal{N}_i$ and $\mathcal{N}_j$ at their observed values $z_{i,j}$, but does not fix connections among units i and j with overlapping neighborhoods.

In line with regression, users are free to choose a fixed design (by fixing both $\boldsymbol{X}$ and $\boldsymbol{Z}$) or a random design (by fixing neither $\boldsymbol{X}$ nor $\boldsymbol{Z}$) or a combination of a fixed design (e.g., fixing $\boldsymbol{X}$) and a random design (e.g., not fixing $\boldsymbol{Z}$). Having said that, fixing $\boldsymbol{X}$ or $\boldsymbol{Z}$ is more restrictive than treating $\boldsymbol{X}$ and $\boldsymbol{Z}$ as random, for two reasons. First, fixing $\boldsymbol{X}$ and $\boldsymbol{Z}$ at their observed values $\boldsymbol{x}$ and $\boldsymbol{z}$ limits conclusions to $\boldsymbol{x}$ and $\boldsymbol{z}$ and does not allow to generalize conclusions to the super population of all possible values of $\boldsymbol{X}$ and $\boldsymbol{Z}$. In other words, random designs afford greater generalizability than fixed designs, at the cost of additional assumptions about the distribution of $\boldsymbol{X}$ and $\boldsymbol{Z}$. Second, fixing $\boldsymbol{X}$ and $\boldsymbol{Z}$ does not provide insight into the mechanism that generates predictors $\boldsymbol{X}$ and connections $\boldsymbol{Z}$.

In the running example, the predictors X_i are indicators of party affiliation, which we consider to be fixed. We fix the predictors X_i at their observed values x_i as follows:

```
R> data.object$set_fix_x(TRUE)
```

Right-hand side of `formula`*: terms*

A comprehensive list of model terms is provided in Supplement A, including attribute-attribute, attribute-connection, and connection-connections terms (e.g., geometrically weighted terms). In addition, users can create custom-built model terms, as described in Section 4.

In the running example, we specify the model as follows:

```
R> formula.iglm <- data.object ~
+                  attribute_y +
+                  attribute_xy +
+                  cov_y(data = gender_attribute) +
+                  cov_y(data = white_attribute) +
+                  edges(mode = "alocal") +
+                  cov_z(data = match_gender, mode = "local") +
+                  cov_z(data = match_race, mode = "local") +
+                  cov_z(data = match_state, mode = "local") +
+                  degrees +
+                  mutual(mode = "local") +
+                  transitive +
+                  spillover_yx(mode = "local") +
+                  spillover_yy(mode = "local")
```

We then create an instance of class `iglm` using

Table 1: Hate speech on X: model terms capturing dependence among attributes (X_i, Y_i) and connections $Z_{i,j}$ in the running example. The term $d_{i,j}(\boldsymbol{z}) \coloneqq \mathbb{I}(\sum_{k \in \mathcal{P} \setminus \{i,j\}} c_{i,k}\, c_{j,k}\, z_{i,k}\, z_{k,j} \geq 1)$ is 1 if units i and j are both connected to at least one unit $k \in \mathcal{N}_i \cap \mathcal{N}_j$ and is 0 otherwise. A comprehensive list of model terms is provided in Supplement A, including attribute-attribute, attribute-connection, and connection-connections terms (e.g., geometrically weighted terms).

Term	Statistic	Weight	Interpretation
1. Attributes	$g_i(\cdot),\ i \in \mathcal{P}$	$\boldsymbol{\theta}_g$	
`attribute_y`	y_i	$\theta_{\mathcal{Y}}$	Intercept for Y_i
`attribute_xy`	$x_i\, y_i$	$\theta_{\mathcal{X},\mathcal{Y}}$	Predictor X_i for Y_i
`cov_y(v)`	$v_i\, y_i$	$\theta_{\mathcal{V},\mathcal{Y}}$	Predictor v_i for Y_i
2. Connections	$h_{i,j}(\cdot),\ (i,j) \in \mathcal{D}$	$\boldsymbol{\theta}_h$	
`edges(mode = "alocal")`	$(1 - c_{i,j})\, z_{i,j}$	$\theta_{\mathcal{Z}}$	Intercept for $Z_{i,j}$
`cov_z(w, mode = "local")`	$c_{i,j}\, w_{i,j}\, z_{i,j}$	$\theta_{\mathcal{W},\mathcal{Z}}$	Predictor $w_{i,j}$ for $Z_{i,j}$
`degrees`	$\sum_{j=1}^{N} z_{i,j}$	$\theta_{\mathcal{O},i}$	Outdegree of unit i
	$\sum_{j=1}^{N} z_{j,i}$	$\theta_{\mathcal{I},i}$	Indegree of unit i
`mutual(mode = "local")`	$\frac{c_{i,j}\, z_{i,j}\, z_{j,i}}{2}$	$\theta_{\mathcal{Z},\mathcal{M}}$	Mutual connection
`transitive`	$c_{i,j}\, d_{i,j}(\boldsymbol{z})\, z_{i,j}$	$\theta_{\mathcal{Z},\mathcal{T}}$	Transitive connection
3. Attributes and Connections	$h_{i,j}(\cdot),\ (i,j) \in \mathcal{D}$	$\boldsymbol{\theta}_h$	
`spillover_yx(mode = "local")`	$c_{i,j}\, x_i\, y_j\, z_{i,j}$	$\theta_{\mathcal{X},\mathcal{Y},\mathcal{Z}}$	Treatment spillover
`spillover_yy(mode = "local")`	$c_{i,j}\, y_i\, y_j\, z_{i,j}$	$\theta_{\mathcal{Y},\mathcal{Y},\mathcal{Z}}$	Outcome spillover

```
R> model.iglm <- iglm(formula = formula.iglm)
```

The definitions of model terms in the running example are presented in Table 1. The term `degrees` captures unobserved heterogeneity in the propensities of units to connect with others by introducing $2\,N$ unit-dependent in- and outdegree terms. Many real-world networks exhibit degree heterogeneity and have long-tailed degree distributions, so capturing degree heterogeneity using `degrees` is advisable. Since maximum likelihood and pseudo-likelihood estimators of degree weights corresponding to isolated units without connections do not exist (e.g., Rinaldo, Fienberg, and Zhou 2009), we exclude isolated units:

```
R> data.object$delete_isolates()
```

Conditional Distributions of Y_i and $Z_{i,j}$ The conditional distributions of outcomes Y_i and connections $Z_{i,j}$ can be derived from the model specified above. We demonstrate in

Section 3.6 that the conditional distribution of $Y_i \in \{0,1\}$ is Bernoulli with mean

$$\mu_{\mathcal{Y},i}(\eta_{\mathcal{Y},i}) \;\coloneqq\; \mathbb{E}_{\eta_{\mathcal{Y},i}}[Y_i \mid (\boldsymbol{X}, \mathbf{Y}_{-i}, \boldsymbol{Z}) = (\boldsymbol{x}, \mathbf{y}_{-i}, \boldsymbol{z})] \;=\; \frac{\exp(\eta_{\mathcal{Y},i})}{1+\exp(\eta_{\mathcal{Y},i})},$$

where

$$\begin{aligned}\eta_{\mathcal{Y},i} \;\coloneqq\; \theta_{\mathcal{Y}} \;+\; \theta_{\mathcal{X},\mathcal{Y}}\, x_i \;+\; \theta_{\mathcal{V},\mathcal{Y},1}\, v_{i,1} \;+\; \theta_{\mathcal{V},\mathcal{Y},2}\, v_{i,2} \;&+\; \theta_{\mathcal{X},\mathcal{Y},\mathcal{Z}} \sum_{j \in \mathcal{P}\setminus\{i\}} c_{i,j}\, x_j\, z_{j,i} \\ &+\; \theta_{\mathcal{Y},\mathcal{Y},\mathcal{Z}} \sum_{j \in \mathcal{P}\setminus\{i\}} c_{i,j}\, y_j\, (z_{i,j} + z_{j,i}),\end{aligned}$$

while the conditional distribution of $Z_{i,j} \in \{0,1\}$ is Bernoulli with mean

$$\mu_{\mathcal{Z},i,j}(\eta_{\mathcal{Z},i,j}) \;\coloneqq\; \mathbb{E}_{\eta_{\mathcal{Z},i,j}}[Z_{i,j} \mid (\boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z}_{-(i,j)}) = (\boldsymbol{x}, \mathbf{y}, \boldsymbol{z}_{-(i,j)})] \;=\; \frac{\exp(\eta_{\mathcal{Z},i,j})}{1+\exp(\eta_{\mathcal{Z},i,j})},$$

where

$$\begin{aligned}\eta_{\mathcal{Z},i,j} \;&\coloneqq\; \theta_{\mathcal{O},i} + \theta_{\mathcal{I},j} + \theta_{\mathcal{Z}}\,(1 - c_{i,j}) \\ &+\; \left[\theta_{\mathcal{V},\mathcal{Z},1}\, \mathbb{I}(v_{i,1} = v_{j,1}) + \theta_{\mathcal{V},\mathcal{Z},2}\, \mathbb{I}(v_{i,2} = v_{j,2}) + \theta_{\mathcal{V},\mathcal{Z},3}\, \mathbb{I}(v_{i,3} = v_{j,3})\right] c_{i,j} \\ &+\; \left[\theta_{\mathcal{Z},\mathcal{M}}\, z_{j,i} \;+\; \theta_{\mathcal{Z},\mathcal{T}}\, \Delta_{\mathcal{Z},i,j}(\boldsymbol{z}) \;+\; \theta_{\mathcal{X},\mathcal{Y},\mathcal{Z}}\, x_i\, y_j \;+\; \theta_{\mathcal{Y},\mathcal{Y},\mathcal{Z}}\, y_i\, y_j\right] c_{i,j}.\end{aligned}$$

Here, $\Delta_{\mathcal{Z},i,j}(\boldsymbol{z}) \coloneqq \sum_{(a,b)\in\mathcal{D}} [d_{a,b}(z_{i,j} = 1, \boldsymbol{z}_{-(i,j)}) - d_{a,b}(z_{i,j} = 0, \boldsymbol{z}_{-(i,j)})]$, where $d_{a,b}(\boldsymbol{z})$ is 1 if units a and b are both connected to one or more units $k \in \mathcal{N}_i \cap \mathcal{N}_j$ and is 0 otherwise.

The conditional distributions of outcomes Y_i and connections $Z_{i,j}$ stated above can be represented by logistic regression models, i.e., GLMs. We demonstrate in Section 3.6 that the GLM representations of these conditionals facilitate interpretation.

3.2. Estimation

To estimate the vector of weights $\boldsymbol{\theta} \in \mathbb{R}^p$, we maximize pseudo-likelihood functions using Minorization-Maximization and Quasi-Newton algorithms; note that likelihood inference based on Markov chain Monte Carlo approximations of the likelihood is feasible when N is small (Hunter and Handcock 2006), but is time-consuming when N is large.

Assuming that both attributes (X_i, Y_i) and connections $Z_{i,j}$ are random, we define the pseudo-loglikelihood function based on an observation $(\boldsymbol{x}, \mathbf{y}, \boldsymbol{z})$ of $(\boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z})$ by

$$\ell(\boldsymbol{\theta}; \boldsymbol{x}, \mathbf{y}, \boldsymbol{z}) \;\coloneqq\; \underbrace{\sum_{i\in\mathcal{P}} \ell_{\mathcal{X},i}(\boldsymbol{\theta}; \boldsymbol{x}, \mathbf{y}, \boldsymbol{z})}_{\textit{predictors}} \;+\; \underbrace{\sum_{i\in\mathcal{P}} \ell_{\mathcal{Y},i}(\boldsymbol{\theta}; \boldsymbol{x}, \mathbf{y}, \boldsymbol{z})}_{\textit{outcomes}} \;+\; \underbrace{\sum_{(i,j)\in\mathcal{D}} \ell_{\mathcal{Z},i,j}(\boldsymbol{\theta}; \boldsymbol{x}, \mathbf{y}, \boldsymbol{z})}_{\textit{connections}}.$$

The components $\ell_{\mathcal{X},i}(\cdot)$, $\ell_{\mathcal{Y},i}(\cdot)$, and $\ell_{\mathcal{Z},i,j}(\cdot)$ of $\ell(\cdot)$ are the contributions of predictors, outcomes, and connections to the pseudo-loglikelihood function $\ell(\cdot)$:

$$\begin{aligned}\ell_{\mathcal{X},i}(\boldsymbol{\theta}; \boldsymbol{x}, \mathbf{y}, \boldsymbol{z}) \;&\coloneqq\; \log f_{\boldsymbol{\theta}}(x_i \mid \boldsymbol{x}_{-i}, \mathbf{y}, \boldsymbol{z}) \\ \ell_{\mathcal{Y},i}(\boldsymbol{\theta}; \boldsymbol{x}, \mathbf{y}, \boldsymbol{z}) \;&\coloneqq\; \log f_{\boldsymbol{\theta}}(y_i \mid \boldsymbol{x}, \mathbf{y}_{-i}, \boldsymbol{z}) \\ \ell_{\mathcal{Z},i,j}(\boldsymbol{\theta}; \boldsymbol{x}, \mathbf{y}, \boldsymbol{z}) \;&\coloneqq\; \log f_{\boldsymbol{\theta}}(z_{i,j} \mid \boldsymbol{x}, \mathbf{y}, \boldsymbol{z}_{-(i,j)}).\end{aligned}$$

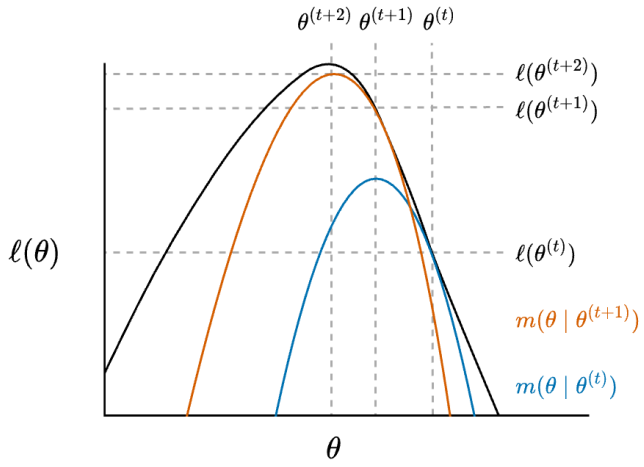

Figure 5: A generic MM algorithm maximizing an objective function $\ell : \mathbb{R} \mapsto \mathbb{R}$ by maximizing a sequence of minorizing functions $m(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})$ and $m(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t+1)})$. The dashed vertical lines represent the starting point $\boldsymbol{\theta}^{(t)}$, the maximizer $\boldsymbol{\theta}^{(t+1)}$ of $m(\cdot \mid \boldsymbol{\theta}^{(t)})$ given $\boldsymbol{\theta}^{(t)}$, and the maximizer $\boldsymbol{\theta}^{(t+2)}$ of $m(\cdot \mid \boldsymbol{\theta}^{(t+1)})$ given $\boldsymbol{\theta}^{(t+1)}$.

If predictors X_i or connections $Z_{i,j}$ are fixed, we drop the corresponding components of $\ell(\cdot)$. The pseudo-loglikelihood function $\ell(\cdot)$ is based on the conditional densities of attributes (X_i, Y_i) and connections $Z_{i,j}$ and is therefore tractable. In addition, the fact that the joint probability density function of $(\boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z})$ stated in Section 3 is a statistical exponential-family density implies that the conditional probability density functions of X_i, Y_i, and $Z_{i,j}$ are exponential-family densities, which in turn implies that $\ell(\cdot)$ is a sum of exponential-family loglikelihood functions. Each of them is concave and twice differentiable (Brown 1986), so maximizing $\ell(\cdot)$ is a concave maximization program.

To estimate $\boldsymbol{\theta}$ when the model includes the term `degrees` with $2\,N$ in- and outdegree weights, one can partition the vector of all weights $\boldsymbol{\theta}$ into the high-dimensional subvector $\boldsymbol{\theta}_1$ of $2\,N$ in- and outdegree weights and the low-dimensional subvector $\boldsymbol{\theta}_2$ consisting of all other weights. Iteration $t+1$ then consists of two steps, given interim estimates $\boldsymbol{\theta}_1^{(t)}$ and $\boldsymbol{\theta}_2^{(t)}$ of $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, respectively:

> **Step 1:** Increase $\ell(\boldsymbol{\theta}_1^{(t)}, \boldsymbol{\theta}_2^{(t)}; \boldsymbol{x}, \mathbf{y}, \boldsymbol{z})$ as a function of $\boldsymbol{\theta}_1^{(t)}$ for fixed $\boldsymbol{\theta}_2^{(t)}$ using a Minorization-Maximization (MM) algorithm, which can be accelerated by a Quasi-Newton (QR) algorithm, and denote the resulting update by $\boldsymbol{\theta}_1^{(t+1)}$.
>
> **Step 2:** Increase $\ell(\boldsymbol{\theta}_1^{(t+1)}, \boldsymbol{\theta}_2^{(t)}; \boldsymbol{x}, \mathbf{y}, \boldsymbol{z})$ as a function of $\boldsymbol{\theta}_2^{(t)}$ for fixed $\boldsymbol{\theta}_1^{(t+1)}$ using a Newton-Raphson algorithm, and denote the resulting update by $\boldsymbol{\theta}_2^{(t+1)}$.

The MM algorithm in Step 1 maximizes $\ell(\cdot)$ by maximizing a well-chosen sequence of minorizing functions. A graphical representation of MM algorithms can be found in Figure 5. The specific MM algorithm is described in Fritz *et al.* (2026). Taken together, Steps 1 and 2

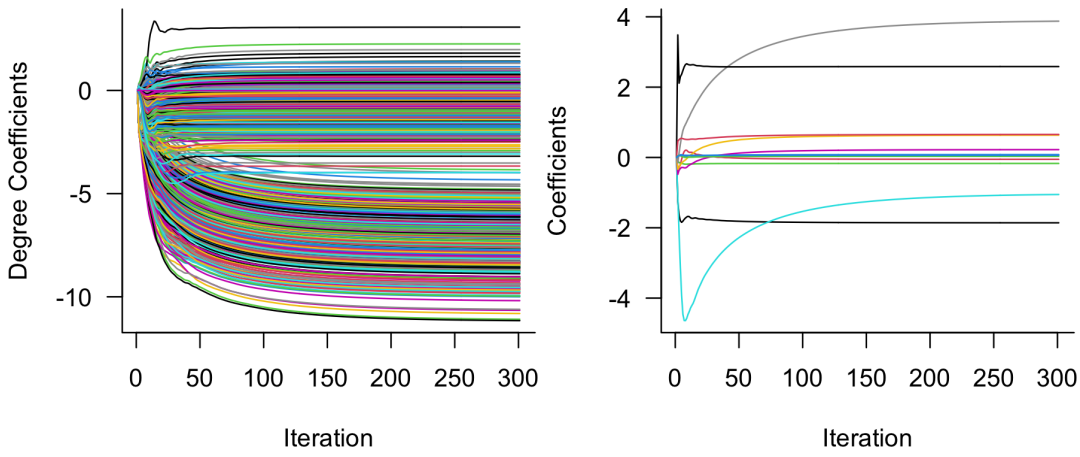

Figure 6: Hate speech on X: trace plots of estimates, including estimates of degree weights (left) and all other weights (right).

increase the concave pseudo-loglikelihood function $\ell(\cdot)$. The resulting pseudo-likelihood estimators come with provable theoretical guarantees—including consistency results and rates of convergence—and are supported by simulation results, as mentioned in Section 1.3.

Example The estimation algorithm can be specified via the object `iglm.control`. In the running example, we use the following options to specify the estimation algorithm:

```
R> control.obj <- control.iglm(max_it = 300)
```

The option `max_it = 300` limits the number of iterations of the estimation algorithm to 300. Additional options allow users to specify which information the estimation algorithm returns. A comprehensive list of available options can be found in the help function of R package **iglm**: see `?control.iglm`.

We use all options specified in `iglm.control` for the `model.iglm` object using

```
R> model.iglm$set_control(control.obj)
```

We then call the function `estimate()` to estimate the model:

```
R> model.iglm$estimate()
```

Output generated by operations on an `iglm` object is stored in `results`, which is a `R6` class containing methods to display diagnostics, including trace plots of estimates:

```
R> model.iglm$results$plot(trace = TRUE)
```

Trace plots of estimates help detect non-convergence of the estimation algorithm. In the running example, the trace plots in Figure 6 do not reveal signs of non-convergence.

3.3. Uncertainty

R package **iglm** quantifies the uncertainty about the maximum pseudo-likelihood estimator $\widehat{\boldsymbol{\theta}}$ based on its exact covariance matrix, without relying on asymptotic normality or other standard asymptotics that may not be applicable to non-standard models for dependent attributes (X_i, Y_i) and connections $Z_{i,j}$.

According to the mean-value theorem for vector-valued functions (Ortega and Rheinboldt 2000, Equations (2) and (3), pp. 68–69), there exist real numbers $t_1, \ldots, t_p \in (0, 1)$ such that the covariance matrix of $\widehat{\boldsymbol{\theta}}$ is

$$\mathbb{V}_{\boldsymbol{\theta}^\star}(\widehat{\boldsymbol{\theta}}) \;=\; \mathbb{V}_{\boldsymbol{\theta}^\star}\left[-\boldsymbol{H}(\widehat{\boldsymbol{\theta}},\, \boldsymbol{\theta}^\star; \boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z})^{-1}\, \nabla_{\boldsymbol{\theta}}\, \ell(\boldsymbol{\theta}; \boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^\star}\right],$$

where $\boldsymbol{\theta}^\star \in \mathbb{R}^p$ is the data-generating vector of weights. The matrix $\boldsymbol{H}(\cdot) \in \mathbb{R}^{p\times p}$ is

$$\boldsymbol{H}(\widehat{\boldsymbol{\theta}},\, \boldsymbol{\theta}^\star; \boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z}) \;\coloneqq\; \begin{pmatrix} g_1'(\boldsymbol{\theta}^\star + t_1\,(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}^\star);\, \boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z}) \\ \vdots \\ g_p'(\boldsymbol{\theta}^\star + t_p\,(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}^\star);\, \boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z}) \end{pmatrix},$$

where $g_k(\cdot)$ is the kth coordinate of $\nabla_{\boldsymbol{\theta}}\, \ell(\cdot)$ and $g_k'(\cdot)$ is the row vector of partial derivatives of $g_k(\cdot)$ with respect to $\boldsymbol{\theta}$ $(k = 1, \ldots, p)$. Since $\boldsymbol{\theta}^\star$ is unknown, we replace $\boldsymbol{\theta}^\star$ by $\widehat{\boldsymbol{\theta}}$, resulting in the approximate covariance matrix

$$\mathbb{V}_{\widehat{\boldsymbol{\theta}}}(\widehat{\boldsymbol{\theta}}) \;=\; \mathbb{V}_{\widehat{\boldsymbol{\theta}}}\left[-\nabla^2_{\boldsymbol{\theta}}\, \ell(\boldsymbol{\theta}; \boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z})\Big|^{-1}_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}}\, \nabla_{\boldsymbol{\theta}}\, \ell(\boldsymbol{\theta}; \boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z})\Big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}}\right].$$

We use Markov chain Monte Carlo simulations of attributes (X_i, Y_i) and connections $Z_{i,j}$ to approximate $\mathbb{V}_{\widehat{\boldsymbol{\theta}}}(\widehat{\boldsymbol{\theta}})$, as described in Section 3.4. The standard errors of $\widehat{\boldsymbol{\theta}}$ are the square roots of the diagonal elements of $\mathbb{V}_{\widehat{\boldsymbol{\theta}}}(\widehat{\boldsymbol{\theta}})$. In the running example, the standard errors of $\widehat{\boldsymbol{\theta}}$ are shown in the second column of the summary reported in Section 3.6.

3.4. Simulation

To quantify the uncertainty about the maximum pseudo-likelihood estimator $\widehat{\boldsymbol{\theta}}$ and assess the model, we simulate attributes (X_i, Y_i) and connections $Z_{i,j}$ using Markov chain Monte Carlo. If both (X_i, Y_i) and $Z_{i,j}$ are random, we can simulate (X_i, Y_i) and $Z_{i,j}$ using Gibbs sampling:

1. Sample $X_i \mid (\boldsymbol{X}_{-i}, \mathbf{Y}, \boldsymbol{Z}) = (\boldsymbol{x}_{-i}, \mathbf{y}, \boldsymbol{z})$ for all units $i \in \mathcal{P}$.
2. Sample $Y_i \mid (\boldsymbol{X}, \mathbf{Y}_{-i}, \boldsymbol{Z}) = (\boldsymbol{x}, \mathbf{y}_{-i}, \boldsymbol{z})$ for all units $i \in \mathcal{P}$.
3. Sample $Z_{i,j} \mid (\boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z}_{-(i,j)}) = (\boldsymbol{x}, \mathbf{y}, \boldsymbol{z}_{-(i,j)})$ for all pairs of units $(i, j) \in \mathcal{D}$.

An alternative to simulating connections $Z_{i,j}$ by Gibbs sampling is Metropolis-Hastings algorithms with TNT (Tie-No-Tie) proposals, which select an unconnected or connected pair of units with equal probability and propose toggling it. Metropolis-Hastings algorithms with TNT proposals are popular in R packages for dependent connections $Z_{i,j}$, including `ergm` (Krivitsky *et al.* 2023) and `Bergm` (Caimo and Friel 2014), and can improve the mixing of Markov chain Monte Carlo algorithms for simulating sparse networks.

In the running example, outcomes Y_i and connections $Z_{i,j}$ are random while predictors X_i are fixed, because the predictors X_i are indicators of party affiliation. We specify Markov chain Monte Carlo algorithms for sampling Y_i and $Z_{i,j}$ as follows:

```
R> sampler.y.obj <- sampler.net.attr(n_proposals = data.object$n_actor * 10)
R> M <- nrow(data.object$overlap)*10
R> sampler.z.obj <- sampler.net.attr(n_proposals = M, tnt = TRUE)
```

The object `sampler.y.obj` specifies a Gibbs sampler for simulating Y_i. The object `sampler.z.obj` specifies a Metropolis-Hastings algorithms with TNT proposals for simulating $Z_{i,j}$. These Markov chain Monte Carlo algorithms can be combined as follows:

```
R> sampler.obj <- sampler.iglm(n_burn_in = 1,
+                               n_simulation = 1,
+                               seed = 123,
+                               sampler_y = sampler.y.obj,
+                               sampler_z = sampler.z.obj)
```

The options `n_burn_in` and `n_simulation` specify the number of burn-in and post-burn-in iterations. Using `sampler.obj`, we can then simulate Y_i and $Z_{i,j}$ as follows:

```
R> model.iglm$set_sampler(sampler.obj)
R> model.iglm$simulate()
```

The resulting samples are saved in `model.iglm$results$samples` as an `iglm.data.list`.

Sometimes, users may wish to simulate from a specified model. To do so, users need to specify the model in Section 3.1 along with the values of all weights. We can specify the values of all weights by specifying the values `coef_degree` of the degree weights (when the term `degrees` is used) and the values `coef` of all other weights:

```
R> model.sim <- iglm(formula = formula.iglm,
+                     sampler = sampler.obj,
+                     coef = model.iglm$coef,
+                     coef_degree = model.iglm$coef_degrees)
R> model.sim$simulate()
```

The function `simulate()` simulates attributes (X_i, Y_i) and connections $Z_{i,j}$. We use simulations to quantify the uncertainty about maximum pseudo-likelihood estimators $\widehat{\boldsymbol{\theta}}$ in Section 3.3. In the above call, we do not specify the number of Markov chain Monte Carlo draws but use the default of 100 draws, starting from the observed data stored in `iglm.data`.

3.5. Assessment

To assess models, R package **iglm** simulates attributes (X_i, Y_i) and connections $Z_{i,j}$ and compares simulated and observed data, helping detect shortcomings of the model. To do so, we call the function `assess()`. Its arguments specify the statistics to be computed.

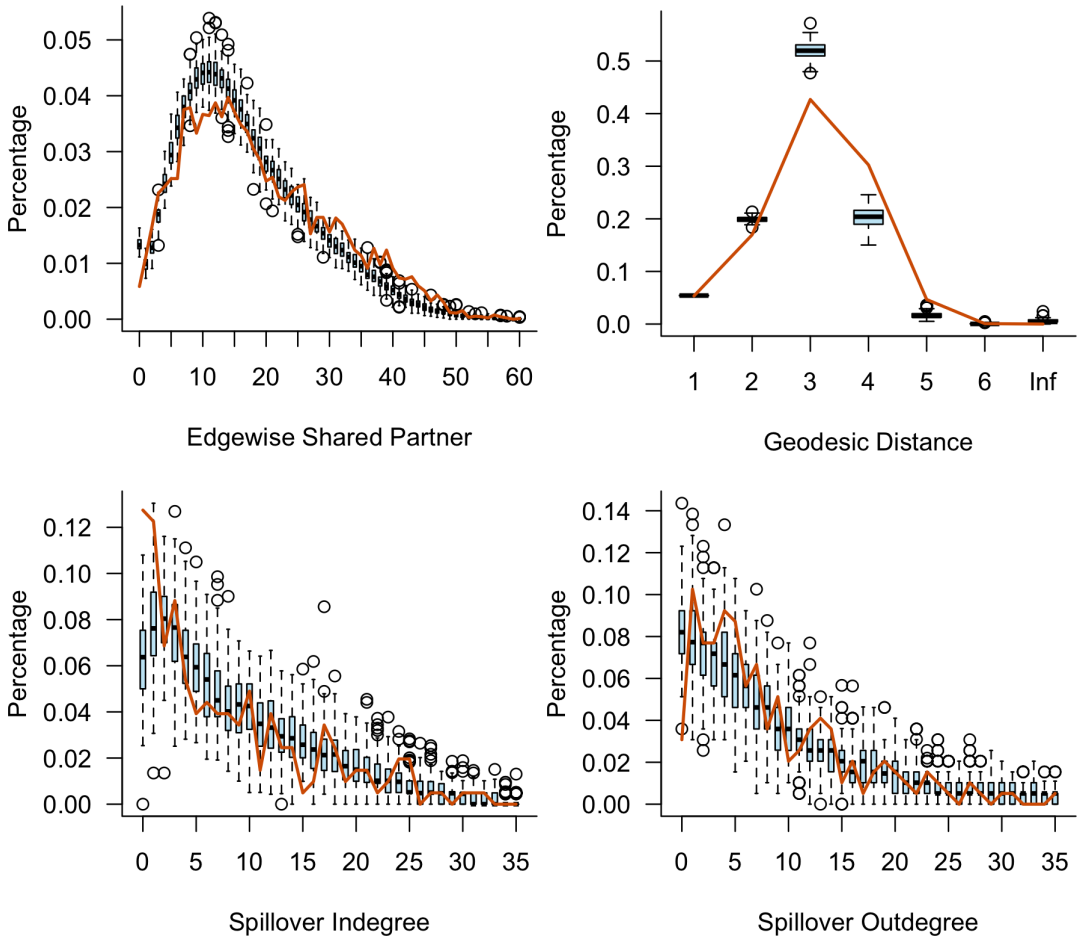

Figure 7: Hate speech on X: assessing the model in terms of the numbers of edgewise shared partners, geodesic distances, and spillover in- and outdegrees described in Section 2.

Example In the running example, we assess the model as follows:

```
R> model.iglm$assess(formula = ~ edgewise_shared_partner_distribution +
+                                geodesic_distances_distribution +
+                                spillover_degree_distribution)
```

The function `assess()` searches in object `model.iglm` for existing simulations. If `model.iglm` contains existing simulations, it will use them. Otherwise, it will simulate data. In either case, it will plot the results, as demonstrated in Figure 7. Figure 7 suggests that model-based predictions match the empirical distributions of the numbers of edgewise shared partners, geodesic distances, and spillover in- and outdegrees.

3.6. Interpretation

Models can be interpreted by inspecting the conditional distributions of attributes (X_i, Y_i) and connections $Z_{i,j}$. The conditional distributions of (X_i, Y_i) and $Z_{i,j}$ can be represented by GLMs as long as the vector-valued functions $g_i(\cdot)$ and $h_{i,j}(\cdot)$ are affine functions of (X_i, Y_i).

The GLM representations of these conditionals facilitate interpretation, because GLMs are widely used in data science and many data scientists are familiar with GLMs.

To demonstrate, let $g_i(\cdot)$ and $h_{i,j}(\cdot)$ be affine functions of (X_i, Y_i) in the sense that there exist vector-valued functions $g_{\mathcal{X},i,k}(\cdot)$, $g_{\mathcal{Y},i,k}(\cdot)$, $h_{\mathcal{X},i,j,k}(\cdot)$, and $h_{\mathcal{Y},i,j,k}(\cdot)$ $(k = 0, 1, 2, 3)$ such that

$$\begin{aligned} g_i(x_i, y_i) &= g_{\mathcal{X},i,0}(y_i) + g_{\mathcal{X},i,1}(y_i)\, x_i \\ &= g_{\mathcal{Y},i,0}(x_i) + g_{\mathcal{Y},i,1}(x_i)\, y_i \\ h_{i,j}(x_i, x_j, y_i, y_j, \boldsymbol{z}) &= h_{\mathcal{X},i,j,0}(y_i, y_j, \boldsymbol{z}) + h_{\mathcal{X},i,j,1}(y_i, y_j, \boldsymbol{z})\, x_i + h_{\mathcal{X},i,j,2}(y_i, y_j, \boldsymbol{z})\, x_j \\ &+ h_{\mathcal{X},i,j,3}(y_i, y_j, \boldsymbol{z})\, x_i\, x_j \\ &= h_{\mathcal{Y},i,j,0}(x_i, x_j, \boldsymbol{z}) + h_{\mathcal{Y},i,j,1}(x_i, x_j, \boldsymbol{z})\, y_i + h_{\mathcal{Y},i,j,2}(x_i, x_j, \boldsymbol{z})\, y_j \\ &+ h_{\mathcal{Y},i,j,3}(x_i, x_j, \boldsymbol{z})\, y_i\, y_j. \end{aligned} \tag{1}$$

We assume that $h_{j,i}(\cdot) = h_{i,j}(\cdot)$ when connections are directed, and define $h_{j,i}(\cdot) \coloneqq 0$ when connections are undirected, because the joint probability density function of $(\boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z})$ stated in Section 3 depends on $h_{i,j}(\cdot)$ but does not depend on $h_{j,i}(\cdot)$ $(i < j)$. Extending Proposition 1 of Fritz *et al.* (2026) from Y_i to X_i and Y_i shows that the conditional distributions of X_i and Y_i can be represented by GLMs with linear predictors

$$\begin{aligned} \eta_{\mathcal{X},i} &\coloneqq \boldsymbol{\theta}^\top \left(g_{\mathcal{X},i,1}(y_i),\ h_{\mathcal{X},i}(x_j, y_i, y_j, \boldsymbol{z})\right) &&=: \boldsymbol{\theta}^\top \Delta_{\mathcal{X},i} \\ \eta_{\mathcal{Y},i} &\coloneqq \boldsymbol{\theta}^\top \left(g_{\mathcal{Y},i,1}(x_i),\ h_{\mathcal{Y},i}(x_i, x_i, y_j, \boldsymbol{z})\right) &&=: \boldsymbol{\theta}^\top \Delta_{\mathcal{Y},i}. \end{aligned} \tag{2}$$

The functions $h_{\mathcal{X},i}(\cdot)$ and $h_{\mathcal{Y},i}(\cdot)$ are defined by

$$\begin{aligned} h_{\mathcal{X},i}(x_j, y_i, y_j, \boldsymbol{z}) &\coloneqq \sum_{j \in \mathcal{P}\setminus\{i\}} \left[h_{\mathcal{X},i,j,1}(\cdot) + h_{\mathcal{X},j,i,2}(\cdot) + h_{\mathcal{X},i,j,3}(\cdot)\, x_j + h_{\mathcal{X},j,i,3}(\cdot)\, x_j\right] \\ h_{\mathcal{Y},i}(x_i, x_i, y_j, \boldsymbol{z}) &\coloneqq \sum_{j \in \mathcal{P}\setminus\{i\}} \left[h_{\mathcal{Y},i,j,1}(\cdot) + h_{\mathcal{Y},j,i,2}(\cdot) + h_{\mathcal{Y},i,j,3}(\cdot)\, y_j + h_{\mathcal{Y},j,i,3}(\cdot)\, y_j\right], \end{aligned}$$

where the arguments of functions on the right-hand side are suppressed for the sake of brevity. The conditional distribution of $Z_{i,j}$ can be represented by a GLM with linear predictor

$$\eta_{\mathcal{Z},i,j} \coloneqq \boldsymbol{\theta}^\top \Big(s(\boldsymbol{x}, \mathbf{y}, z_{i,j} = 1, \boldsymbol{z}_{-(i,j)}) - s(\boldsymbol{x}, \mathbf{y}, z_{i,j} = 0, \boldsymbol{z}_{-(i,j)})\Big) =: \boldsymbol{\theta}^\top \Delta_{\mathcal{Z},i,j}, \tag{3}$$

where

$$s(\boldsymbol{x}, \mathbf{y}, z_{i,j}, \boldsymbol{z}_{-(i,j)}) \coloneqq \sum_{(a,b) \in \mathcal{D}} h_{a,b}(x_a, x_b, y_a, y_b, z_{i,j}, \boldsymbol{z}_{-(i,j)}).$$

We demonstrate how the conditional distribution of Y_i can be represented by a GLM in the running example. First, the conditional distribution of $Y_i \in \{0, 1\}$ is Bernoulli with mean

$$\mu_{\mathcal{Y},i}(\eta_{\mathcal{Y},i}) \coloneqq \mathbb{E}_{\eta_{\mathcal{Y},i}}[Y_i \mid (\boldsymbol{X}, \mathbf{Y}_{-i}, \boldsymbol{Z}) = (\boldsymbol{x}, \mathbf{y}_{-i}, \boldsymbol{z})] = \frac{\exp(\eta_{\mathcal{Y},i})}{1 + \exp(\eta_{\mathcal{Y},i})}.$$

Second, the linear predictor $\eta_{\mathcal{Y},i}$ can be derived by starting with the first term in the formula `formula.iglm` stated in Section 3.1 and adding the other terms one by one:

1. `attribute_y` implies that $g_i(\cdot)$ includes coordinate y_i. Since the term y_i with weight $\theta_{\mathcal{Y}}$ is a linear function of y_i, `attribute_y` adds $\theta_{\mathcal{Y}}$ to linear predictor $\eta_{\mathcal{Y},i}$.

2. Adding `attribute_xy` adds coordinate $x_i\, y_i$ to $g_i(\cdot)$. Since the term $x_i\, y_i$ with weight $\theta_{\mathcal{X},\mathcal{Y}}$ is a linear function of y_i, `attribute_xy` adds $\theta_{\mathcal{X},\mathcal{Y}}\, x_i$ to linear predictor $\eta_{\mathcal{Y},i}$.

3–4. Adding `cov_y` with predictors `gender_attribute` and `white_attribute` adds coordinates $v_{i,1}\, y_i$ and $v_{i,2}\, y_i$ to $g_i(\cdot)$. The terms $v_{i,1}\, y_i$ and $v_{i,2}\, y_i$ with weights $\theta_{\mathcal{V},\mathcal{Y},1}$ and $\theta_{\mathcal{V},\mathcal{Y},2}$ are linear functions of y_i and add $\theta_{\mathcal{V},\mathcal{Y},1}\, v_{i,1} + \theta_{\mathcal{V},\mathcal{Y},2}\, v_{i,2}$ to linear predictor $\eta_{\mathcal{Y},i}$.

5. Adding `spillover_yx` implies that $h_{i,j}(\cdot)$ and $h_{j,i}(\cdot)$ consist of $c_{i,j}\, x_i\, y_j\, z_{i,j}$ and $c_{j,i}\, x_j\, y_i\, z_{j,i}$, each of them with weight $\theta_{\mathcal{X},\mathcal{Y},\mathcal{Z}}$. Both terms are linear functions of y_i, adding $\theta_{\mathcal{X},\mathcal{Y},\mathcal{Z}} \sum_{j\in\mathcal{P}\setminus\{i\}} c_{i,j}\, x_j\, z_{j,i}$ to linear predictor $\eta_{\mathcal{Y},i}$; note that $c_{i,j} = c_{j,i}$.

6. Adding `spillover_yy` adds $c_{i,j}\, y_i\, y_j\, z_{i,j}$ and $c_{j,i}\, y_j\, y_i\, z_{j,i}$ to $h_{i,j}(\cdot)$ and $h_{j,i}(\cdot)$, each with weight $\theta_{\mathcal{Y},\mathcal{Y},\mathcal{Z}}$. Both are linear functions of y_i, adding $\theta_{\mathcal{Y},\mathcal{Y},\mathcal{Z}} \sum_{j\in\mathcal{P}\setminus\{i\}} c_{i,j}\, y_j\, (z_{i,j}+z_{j,i})$ to linear predictor $\eta_{\mathcal{Y},i}$.

All other terms in `formula.iglm` are not functions of y_i and hence do not affect the conditional distribution of Y_i. Upon collecting terms, the linear predictor turns out to be

$$\eta_{\mathcal{Y},i} \;\coloneqq\; \theta_{\mathcal{Y}} \;+\; \theta_{\mathcal{X},\mathcal{Y}}\, x_i \;+\; \theta_{\mathcal{V},\mathcal{Y},1}\, v_{i,1} \;+\; \theta_{\mathcal{V},\mathcal{Y},2}\, v_{i,2} \;+\; \theta_{\mathcal{X},\mathcal{Y},\mathcal{Z}} \sum_{j\in\mathcal{P}\setminus\{i\}} c_{i,j}\, x_j\, z_{j,i} \;+\; \theta_{\mathcal{Y},\mathcal{Y},\mathcal{Z}} \sum_{j\in\mathcal{P}\setminus\{i\}} c_{i,j}\, y_j\, (z_{i,j} + z_{j,i}).$$

Example Using the function `summary()`, we obtain the following maximum pseudo-likelihood estimate $\widehat{\boldsymbol{\theta}}$ and standard errors:

```
R> model.iglm$summary()
iglm object
------------------------------------------------------------------------------
Results:

                                              Estimate    S.E. t-value Pr(>|t|)
attribute_y                                    -1.8587  0.3314   -5.61  <0.0001
attribute_xy                                   -0.0533  0.2242   -0.24     0.81
cov_y(data = gender_attribute)                 -0.1703  0.2359   -0.72     0.47
cov_y(data = white_attribute)                   0.0806  0.2274    0.35     0.72
edges(mode = 'alocal')                         -1.0531  0.0039 -272.44  <0.0001
cov_z(data = match_gender, mode = 'local')      0.2226  0.0145   15.38  <0.0001
cov_z(data = match_race, mode = 'local')        0.6343  0.0095   66.99  <0.0001
cov_z(data = match_state, mode = 'local')       3.8760  0.0268  144.75  <0.0001
mutual(mode = 'local')                          2.5847  0.0502   51.46  <0.0001
transitive                                      0.6546  0.0169   38.74  <0.0001
spillover_yx(mode = 'local')                    0.0325  0.0335    0.97     0.33
spillover_yy(mode = 'local')                    0.0626  0.0124    5.06  <0.0001
```

```
Time for estimation: 20 mins

Degree Parameters:
  Outdegrees:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -11.2    -7.9    -6.9    -7.1    -6.2    -3.8

  Indegrees:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -3.985  -1.104  -0.563  -0.613  -0.027   3.069
```

Since there are $2\,N$ in- and outdegree weights, the function `summary()` does not print estimates of all $2\,N$ in- and outdegree weights. Instead, it prints a summary of the $2\,N$ estimates. Users can access all $2\,N$ estimates using `model.iglm$coef_degrees`.

The conditional distribution of outcome $Y_i \in \{0, 1\}$ is Bernoulli with mean

$$\mu_{\mathcal{Y},i}(\widehat{\eta}_{\mathcal{Y},i}) \;\coloneqq\; \mathbb{E}_{\widehat{\eta}_{\mathcal{Y},i}}[Y_i \mid (\boldsymbol{X}, \mathbf{Y}_{-i}, \boldsymbol{Z}) = (\boldsymbol{x}, \mathbf{y}_{-i}, \boldsymbol{z})] \;=\; \frac{\exp(\widehat{\eta}_{\mathcal{Y},i})}{1 + \exp(\widehat{\eta}_{\mathcal{Y},i})},$$

where

$$\begin{aligned} \widehat{\eta}_{\mathcal{Y},i} \;&\coloneqq\; -1.859 \;-\; .053\, x_i \;-\; .170\, v_{i,1} \;+\; .081\, v_{i,2} \\ &+\; .033 \sum_{j \in \mathcal{P} \setminus \{i\}} c_{i,j}\, x_j\, z_{j,i} \;+\; .063 \sum_{j \in \mathcal{P} \setminus \{i\}} c_{i,j}\, y_j\, (z_{i,j} + z_{j,i}). \end{aligned}$$

These results indicate that demographic background in terms of party affiliation, gender, and race does not help predict hate speech. The treatment spillover effect is not significant at conventional significance levels (e.g., .05), whereas the outcome spillover effect is significant. To interpret the outcome spillover term, consider any legislator i. Then each legislator j who is close to i ($c_{i,j} = 1$), connected to i (either $z_{i,j} = 1$ or $z_{j,i} = 1$), and uses hate speech ($y_j = 1$) adds .063 to the log odds of the conditional probability that i uses hate speech ($Y_i = 1$).

The conditional distribution of connection $Z_{i,j} \in \{0, 1\}$ is Bernoulli with mean

$$\mu_{\mathcal{Z},i,j}(\widehat{\eta}_{\mathcal{Z},i,j}) \;\coloneqq\; \mathbb{E}_{\widehat{\eta}_{\mathcal{Z},i,j}}[Z_{i,j} \mid (\boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z}_{-(i,j)}) = (\boldsymbol{x}, \mathbf{y}, \boldsymbol{z}_{-(i,j)})] \;=\; \frac{\exp(\widehat{\eta}_{\mathcal{Z},i,j})}{1 + \exp(\widehat{\eta}_{\mathcal{Z},i,j})},$$

where

$$\begin{aligned} \widehat{\eta}_{\mathcal{Z},i,j} \;&\coloneqq\; \widehat{\theta}_{\mathcal{O},i} + \widehat{\theta}_{\mathcal{I},j} - 1.053\,(1 - c_{i,j}) \\ &+\; \left[.223\, \mathbb{I}(v_{i,1} = v_{j,1}) + .634\, \mathbb{I}(v_{i,2} = v_{j,2}) + 3.876\, \mathbb{I}(v_{i,3} = v_{j,3})\right] c_{i,j} \\ &+\; \left[2.585\, z_{j,i} + .655\, \Delta_{\mathcal{Z},i,j}(\boldsymbol{z}) + .033\, x_i\, y_j + .063\, y_i\, y_j\right] c_{i,j}. \end{aligned}$$

The estimates $\widehat{\theta}_{\mathcal{O},i}$ and $\widehat{\theta}_{\mathcal{I},j}$ can be extracted from `model.iglm$coef_degrees`. The above results indicate a tendency to connect with others who are similar in terms of gender, race, state, and hate speech, in addition to effects of mutuality (reciprocity) and transitive closure (transitivity), which aligns with other findings in network analysis.

4. Advanced Topics

We discuss advanced topics, including model comparisons in Section 4.1 and custom-built model terms in Section 4.2.

4.1. Model Comparison

While model selection methods with theoretical guarantees for dependent attributes (X_i, Y_i) and connections $Z_{i,j}$ have not been developed, it is possible to compare models based on predictions. We assess the added value of the joint probability model for $(\boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z})$ specified in Section 3.1 over GLMs for $Y_i \mid X_i = x_i$ and $Z_{i,j} \mid (X_i, X_j) = (x_i, x_j)$ based on predictions. We specify and estimate a simple GLM for $Y_i \mid X_i = x_i$ and $Z_{i,j} \mid (X_i, X_j) = (x_i, x_j)$:

```
R> formula.glm <- data.object ~
+                  attribute_y +
+                  attribute_xy +
+                  cov_y(data = white_attribute) +
+                  cov_y(data = gender_attribute) +
+                  edges(mode = "alocal") +
+                  edges(mode = "local")  +
+                  cov_z(data = match_gender, mode = "local") +
+                  cov_z(data = match_race, mode = "local") +
+                  cov_z(data = match_state, mode = "local")
R> model.glm <- iglm(formula = formula.glm,
+                     control = control.obj,
+                     name = "glm")
R> model.glm$estimate()
```

The model specified above implies that $Y_i \mid X_i = x_i \overset{\text{ind}}{\sim} \text{Bernoulli}(\mu_i)$ and $Z_{i,j} \mid (X_i, X_j) = (x_i, x_j) \overset{\text{ind}}{\sim} \text{Bernoulli}(\mu_{i,j})$, where

$$\log \frac{\mu_i}{1-\mu_i} \quad := \quad \theta_{\mathcal{Y}} + \theta_{\mathcal{X},\mathcal{Y}}\, x_i + \theta_{\mathcal{V},\mathcal{Y},1}\, v_{i,1} + \theta_{\mathcal{V},\mathcal{Y},2}\, v_{i,2}$$

and

$$\log \frac{\mu_{i,j}}{1-\mu_{i,j}} \quad := \quad \theta_{\mathcal{Z},1}\, c_{i,j} + \theta_{\mathcal{Z},2}\,(1 - c_{i,j}) + \sum_{k=1}^{3} \theta_{\mathcal{V},\mathcal{Z},k}\, \mathbb{I}(v_{i,k} = v_{j,k}).$$

We refer to the above model as a GLM, and to the model specified in Section 3.1 as a IGLM. The diagnostic framework introduced in Section 3.5 enables a visual comparison of models based on predictions of attributes using the function `assess()`, suppressing the default graphical output by setting `plot = FALSE`:

```
R> model.iglm$assess(formula = ~ spillover_degree_distribution +
+                                edgewise_shared_partner_distribution,
+                                plot = FALSE)
R> model.glm$assess(formula = ~  spillover_degree_distribution +
+                                edgewise_shared_partner_distribution,
+                                plot = FALSE)
```

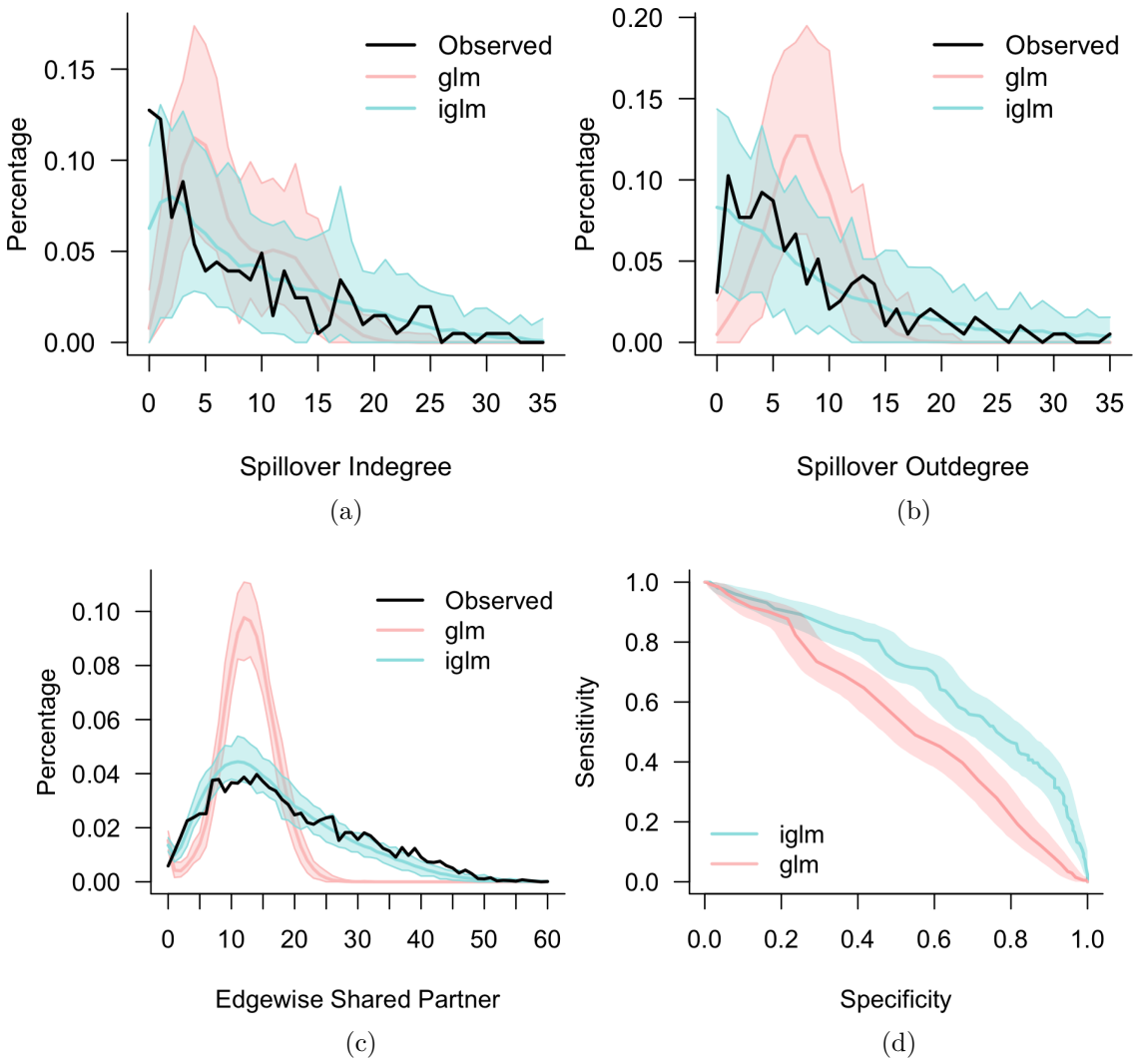

Figure 8: Hate speech on X: model assessment based on predictions. Plots (a)–(c) compare regression models without interference (GLM) and regression models with interference (IGLM). Plot (d) shows the Receiver Operating Characteristic (ROC) curves, illustrating the trade-off between the true and false positive rate for outcomes Y_i under the GLM and IGLM. The shaded areas in plots (a)–(c) represent the minima and maxima over all simulations, while the shaded areas in plot (d) represent a 95% confidence region.

Using the dots (`...`) mechanism of the `plot()` method of the `results` object, users can compare predictions:

```
R> model.glm$results$plot(model_assessment = TRUE,
+                         "iglm" = model.iglm$results$model_assessment)
```

The results are shown in Figure 8. We interpret these results below, but we first discuss conditional and marginal predictions of outcomes Y_i.

Conditional and Marginal Predictions

The method `predict` can calculate conditional and marginal predictions of outcomes Y_i based on approximations of conditional expectations (holding all other variables constant) and marginal expectations (averaging simulations from the model). To compare observed and predicted quantities, we leverage established diagnostic frameworks for GLMs:

1. $Y_i \in \{0, 1\}$: Receiver Operating Characteristic (ROC) or Precision-Recall curves implemented in R package **pROC** (Robin, Turck, Hainard, Tiberti, Lisacek, Sanchez, and Müller 2011).

2. $Y_i \in \{0, 1, \ldots\}$: rootograms available in R package **vcd** (Meyer, Zeileis, Hornik, and Friendly 2024).

3. $Y_i \in \mathbb{R}$: scatterplots comparing predicted and observed outcomes.

Example We compute marginal predictions of attributes (X_i, Y_i) using the `predict` method:

```
R> prediction.iglm <- model.iglm$predict(variant = "marginal")
R> prediction.glm <- model.glm$predict(variant = "marginal")
```

The resulting list provides the marginal probabilities of $Y_i = 1$ and $Z_{i,j} = 1$ under IGLM and GLM. We demonstrate how R package `pROC` can be used to compare predictions of outcomes Y_i based on the two models using ROC curves:

```
R> library(pROC)
R> roc.iglm <-  roc(prediction.iglm$y$target, prediction.iglm$y$prediction)
R> ciobj.iglm <- ci.se(roc.iglm, specificities = seq(0, 1, l = 100))
R> roc.glm <- roc(prediction.glm$y$target, prediction.glm$y$prediction)
R> ciobj.glm <- ci.se(roc.glm, specificities = seq(0, 1, l = 100))
R> plot(roc.iglm$specificities, roc.iglm$sensitivities,  col = "#9AE1E3",
+       type = "l", lwd = 2, bty ="l", lat = 1,
+       xlab = "Specificity",ylab = "Sensitivity")
R> plot(ciobj.iglm, type = "shape", col = "#9AE1E366", border = NA,
+       no.roc = TRUE)
R> lines(roc.glm$specificities, roc.glm$sensitivities,  col = "#FDC7C5",
+        type = "l", lwd = 2, bty ="l",
+        xlab = "Specificity",ylab = "Sensitivity")
R> plot(ciobj.glm, type = "shape", col = "#FF000022", border = NA,
+       no.roc = TRUE)
R> legend("bottomleft",legend = c("iglm", "glm"),lty = c(1,1),lwd = c(2,2),
+         col = c("#9AE1E3", "#FDC7C5"), bty = "n")
```

The results are shown in Figure 8. These results contain two important lessons: First, the fact that model-based predictions of the network based on IGLMs are superior to those based on GLMs suggests that unobserved heterogeneity in the connection propensities of units should be captured along with transitive closure, using model terms such as `degrees` and `transitive`. Second, the IGLM is superior to GLM in terms of model-based predictions of outcomes, suggesting that models should include spillover terms to better predict outcomes.

4.2. User-Defined Model Terms

Users can add custom-built model terms to the list of available model terms stated in Supplement A, which increases the versatility of R package **iglm**. Since the estimation and simulation algorithms in **iglm** are implemented in C++, custom-built model terms need to be implemented in C++. To do so, users first need to call the function

```
R> create_userterms_skeleton()
```

This function creates an R package called **iglm.userterms**, which allows users to add custom-built model terms that can be accessed by **iglm**. To add a custom-built model term to R package **iglm.userterms**, users can follow a simple recipe consisting of three steps:

1. Decompose $g_i(\cdot)$ and $h_{i,j}(\cdot)$ in accordance with Equation (1).
2. Specify the change statistics $\Delta_{\mathcal{X},i}$, $\Delta_{\mathcal{Y},i}$, and $\Delta_{\mathcal{Z},i,j}$ defined in Equations (2) and (3).
3. Write a C++ function to compute the change statistics $\Delta_{\mathcal{X},i}$, $\Delta_{\mathcal{Y},i}$, and $\Delta_{\mathcal{Z},i,j}$ and register it under the name `my_term`. Store the C++ function in a file with extension `.cpp` and add it to the `src` folder of **iglm.userterms**.
4. Write a R function called `InitIglmTerm.my_term` to check the arguments of the C++ function; note that the extension `my_term` of the R function needs to match the name under which the C++ function is registered. Store the R function in the `R` folder of **iglm.userterms**.

To demonstrate, we implement the treatment spillover term

$$c_{i,j}\,(x_i\,y_j + x_j\,y_i)\,z_{i,j} \quad = \quad c_{i,j}\,x_i\,y_j\,z_{i,j} + c_{i,j}\,x_j\,y_i\,z_{i,j} \tag{4}$$

assuming that connections are undirected, so that $z_{i,j} = z_{j,i}$ for all $i < j$. We walk users through each step of the recipe sketched above:

1. The treatment spillover term in Equation (4) is included in the function $h_{i,j}(\cdot)$, which can be decomposed along the lines of Equation (1) as follows:

$$\begin{aligned}
h_{\mathcal{X},i,j,0}(y_i, y_j, \boldsymbol{z}) &\coloneqq 0, & h_{\mathcal{X},i,j,1}(y_i, y_j, \boldsymbol{z}) &\coloneqq c_{i,j}\,y_j\,z_{i,j} \\
h_{\mathcal{X},i,j,2}(y_i, y_j, \boldsymbol{z}) &\coloneqq c_{i,j}\,y_i\,z_{i,j}, & h_{\mathcal{X},i,j,3}(y_i, y_j, \boldsymbol{z}) &\coloneqq 0 \\
h_{\mathcal{Y},i,j,0}(x_i, x_j, \boldsymbol{z}) &\coloneqq 0, & h_{\mathcal{Y},i,j,1}(x_i, x_j, \boldsymbol{z}) &\coloneqq c_{i,j}\,x_j\,z_{i,j} \\
h_{\mathcal{Y},i,j,2}(x_i, x_j, \boldsymbol{z}) &\coloneqq c_{i,j}\,x_i\,z_{i,j}, & h_{\mathcal{Y},i,j,3}(x_i, x_j, \boldsymbol{z}) &\coloneqq 0,
\end{aligned}$$

   assuming that $i < j$; note that $h_{\mathcal{X},i,j,k}(\cdot) \coloneqq 0$ and $h_{\mathcal{Y},i,j,k}(\cdot) \coloneqq 0$ $(k = 0, 1, 2, 3)$ for all $i > j$, because we focus on undirected connections.

2. The corresponding change statistics $\Delta_{\mathcal{X},i}$, $\Delta_{\mathcal{Y},i}$, and $\Delta_{\mathcal{Z},i,j}$ are

$$\Delta_{\mathcal{X},i} \coloneqq \sum_{j \in \mathcal{P}\setminus\{i\}} c_{i,j}\,y_j\,z_{i,j}, \quad \Delta_{\mathcal{Y},i} \coloneqq \sum_{j \in \mathcal{P}\setminus\{i\}} c_{i,j}\,x_j\,z_{i,j}, \quad \Delta_{\mathcal{Z},i,j} \coloneqq c_{i,j}\,(x_i\,y_j + y_i\,x_j).$$

3. We write the C++ function `my_stat_spillover` to compute the change statistics $\Delta_{\mathcal{X},i}$, $\Delta_{\mathcal{Y},i}$, and $\Delta_{\mathcal{Z},i,j}$. The C++ function `my_stat_spillover` relies on methods of C++ class `XYZ_class`, which is the C++ analog of the class `iglm` described in Section 3.1. We refer to Section B for details on how one can use `XYZ_class`. The change statistic $\Delta_{\mathcal{X},i}$ is computed by the following C++ function on lines 10–15, $\Delta_{\mathcal{Y},i}$ is computed on lines 16–21, and $\Delta_{\mathcal{Z},i,j}$ is computed on lines 22–29:

```
1  double my_stat_spillover(const XYZ_class &object,
2                           const int &unit_i,
3                           const int &unit_j,
4                           const arma::mat &data,
5                           const double &type,
6                           const std::string &mode,
7                           const bool &is_full_neighborhood)
8  {
9    double res = 0.0;
10   if (mode == "x") { // x_i from 0 -> 1
11     const auto& connections_of_i = object.adj_list_nb.at(unit_i);
12     for (const int &k : connections_of_i) {
13       res += object.y_attribute.get_val(k);
14     }
15   }
16   else if (mode == "y") { // y_i from 0 -> 1
17     const auto& connections_of_i = object.adj_list_nb.at(unit_i);
18     for (const int &k : connections_of_i) {
19       res += object.x_attribute.get_val(k);
20     }
21   }
22   else { // z_ij from 0 -> 1
23     if (object.get_val_overlap(unit_i, unit_j)) {
24       res = (object.x_attribute.get_val(unit_i) *
25              object.y_attribute.get_val(unit_j)) +
26             (object.x_attribute.get_val(unit_j) *
27              object.y_attribute.get_val(unit_i));
28     }
29   }
30   return res;
31 }
32 EFFECT_REGISTER("my_spillover", ::my_stat_spillover, "my_spillover", 0);
```

   The function `EFFECT_REGISTER` registers the term under the name `my_spillover` on line 32. The last argument of `EFFECT_REGISTER` specifies the value of the statistic when no connections are observed, which is 0. We store the C++ function `xyz_stat_my_spillover` in the `src` folder of **iglm.userterms**.

4. Since the C++ function was registered under the name `my_spillover`, we define the corresponding R function `InitIglmTerm.my_spillover` as follows:

```
R> InitIglmTerm.my_spillover <- function(data_object, arglist, ...) {
+    arglist <- iglm:::check.IglmTerm(data_object, arglist,
+                                     directed = FALSE)
+    list(
+         term_name = "my_spillover",
+         coef_name = arglist$label
+    )
+  }
```

We store the R function `InitIglmTerm.my_spillover` in the R folder of **iglm.userterms**.

The final step is to compile `iglm.userterms` and load it after `iglm`:

```
R> library(iglm)
R> library(iglm.userterms)
```

Users who are interested in adding custom-built model terms can inspect the `C++` file `change_statistics.cpp` in the C++ source folder of **iglm**, which contains `C++` implementations of all model terms listed in Supplement A.

5. Application: Real-Valued Outcomes

In addition to the running example with binary outcomes $Y_i \in \{0, 1\}$, we provide an example with real-valued outcomes $Y_i \in \mathbb{R}$. To do so, we use the Copenhagen network study concerned with communications among $N = 409$ students at a university in Copenhagen, Denmark over 28 days. The data collection procedure is described in Sapiezynski, Stopczynski, Lassen, and Lehmann (2019) and involved recording interaction metrics, including Bluetooth-based proximity scans and self-reported friendships, where $Z_{i,j} := 1$ indicates that i considers j to be a friend or j considers i to be a friend, and $Z_{i,j} := 0$ otherwise. In other words, the network is undirected, so $Z_{i,j} = Z_{j,i}$ for all pairs of students i and j. The fixed predictor $X_i \in \{0, 1\}$ encodes gender (1 if female and 0 otherwise). The outcome $Y_i \in \mathbb{R}$ represents the log-transformed total call duration $Y_i := \log(t_i)$, where t_i is the total number of minutes spent on phone calls; note that $t_i = 0$ is impossible because the availability of call information was one of the inclusion criteria in the study. The neighborhood of student i is defined as

$$\mathcal{N}_i \;:=\; \{j \in \mathcal{P} \setminus \{i\} : \; j \textit{ was detected by } i\textit{'s Bluetooth device for at least } 24 \textit{ hours}\}.$$

The preprocessed data object is included in R package `iglm` as an example data set, and we can load and plot it as follows:

```
R> data("copenhagen")
R> set.seed(321)
R> copenhagen$plot()
```

The function `plot()` plots the network in Figure 9. Since $Y_i \in \mathbb{R}$, `attribute_y` is set to `"normal"`, and the conditional variance of Y_i is estimated by the empirical variance:

```
R> copenhagen$set_scale_y(var(copenhagen$y_attribute))
```

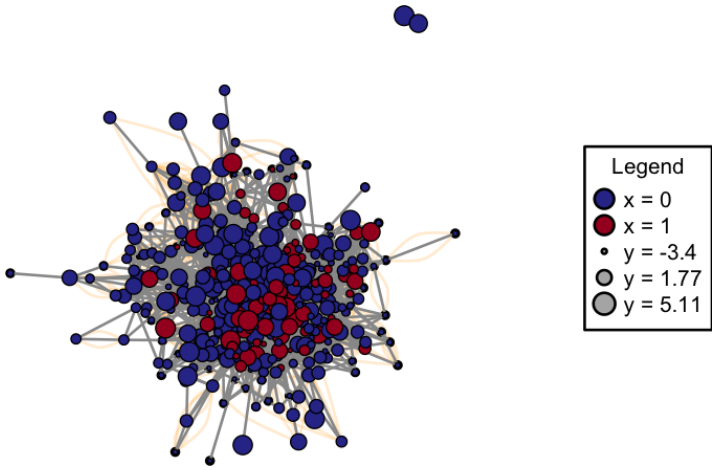

Figure 9: Copenhagen network study: The colors and sizes of the circles represent the gender X_i and log-transformed call time Y_i of students i, respectively.

Descriptive statistics can be obtained as follows:

```
R> copenhagen
iglm.data object
  units                          : 409
  directed                       : FALSE
  edges (fixed = FALSE)          : 2517
  neighborhood edges             : 744

Attribute summaries
  x_attribute (fixed = TRUE)   : binomial 1s=90, 0s=319, P(1)=0.220
  y_attribute                  : normal mean=1.334, sd=1.777, scale= 3.157
```

We then specify the model and estimate it:

```
R> formula.norm <- copenhagen ~
+                  attribute_y +
+                  attribute_xy +
+                  degrees +
+                  edges(mode = "alocal") +
+                  transitive +
+                  spillover_xy +
+                  spillover_yy
R> model.norm <- iglm(formula = formula.norm, control = control.obj)
R> model.norm$estimate()
```

The results are:

```
R> model.norm$summary()
iglm object
-----------------------------------------------------
Results:

                        Estimate    S.E. t-value Pr(>|t|)
attribute_y               1.0879  0.1104     9.8 < 0.0001
attribute_xy              0.4205  0.2071     2.0  0.04228
edges(mode = 'alocal')   -2.7644  0.0012 -2304.0 < 0.0001
transitive                1.9037  0.1741    10.9 < 0.0001
spillover_xy             -0.2327  0.1244    -1.9  0.06126
spillover_yy              0.3269  0.0899     3.6  0.00028

Time for estimation: 5.6 secs

Degree Parameters:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -3.30   -1.24   -0.76   -0.74   -0.19    1.53
```

The conditional distribution of outcome $Y_i \in \mathbb{R}$ is Normal with mean

$$\mu_{\mathrm{y},i}(\widehat{\eta}_{\mathrm{y},i}) \;\coloneqq\; 1.088 + .421\, x_i - .233 \sum_{j \in \mathcal{P}\setminus\{i\}} c_{i,j}\, x_j\, z_{i,j} + .327 \sum_{j \in \mathcal{P}\setminus\{i\}} c_{i,j}\, y_j\, z_{i,j},$$

where $\mu_{\mathrm{y},i}(\widehat{\eta}_{\mathrm{y},i}) = \widehat{\eta}_{\mathrm{y},i}$ because $Y_i \in \mathbb{R}$ and the linear predictor differs from the linear predictor in the running example because $z_{i,j} = z_{j,i}$ and $h_{j,i}(\cdot) \coloneqq 0$ $(i < j)$. While the negative treatment spillover effect is not significant, the positive outcome spillover effect is significant at conventional significance levels (e.g., .05). In other words, the more time i's friends j spends on phone calls, the more time i is expected to spend on phone calls.

The conditional distribution of connection $Z_{i,j} \in \{0, 1\}$ is Bernoulli with mean

$$\mu_{\mathrm{z},i,j}(\widehat{\eta}_{\mathrm{z},i,j}) \;\coloneqq\; \mathbb{E}_{\widehat{\eta}_{\mathrm{z},i,j}}[Z_{i,j} \mid (\boldsymbol{X}, \mathbf{Y}, \boldsymbol{Z}_{-(i,j)}) = (\boldsymbol{x}, \mathbf{y}, \boldsymbol{z}_{-(i,j)})] \;=\; \frac{\exp(\widehat{\eta}_{\mathrm{z},i,j})}{1 + \exp(\widehat{\eta}_{\mathrm{z},i,j})},$$

where

$$\widehat{\eta}_{\mathrm{z},i,j} \coloneqq \widehat{\theta}_{\mathrm{z},i} + \widehat{\theta}_{\mathrm{z},j} - 2.764\,(1 - c_{i,j}) + c_{i,j}\,\left[1.904\,\Delta_{\mathrm{z},i,j}(\boldsymbol{z}) - .233\,(x_i\, y_j + x_j\, y_i) + .327\, y_i\, y_j\right],$$

and $\widehat{\theta}_{\mathrm{z},i}$ and $\widehat{\theta}_{\mathrm{z},j}$ can be interpreted as the propensities of students i and j to connect with others. Here, too, there is strong evidence for transitive closure among friends: More often than not, a friend of a friend is a friend.

Finally, we assess the model by comparing simulated and observed outcomes Y_i as well as spillover in- and outdegrees and geodesic distance distributions:

```
R> model.norm$assess(formula = ~ y_distribution +
+                                geodesic_distances_distribution +
+                                spillover_degree_distribution)
```

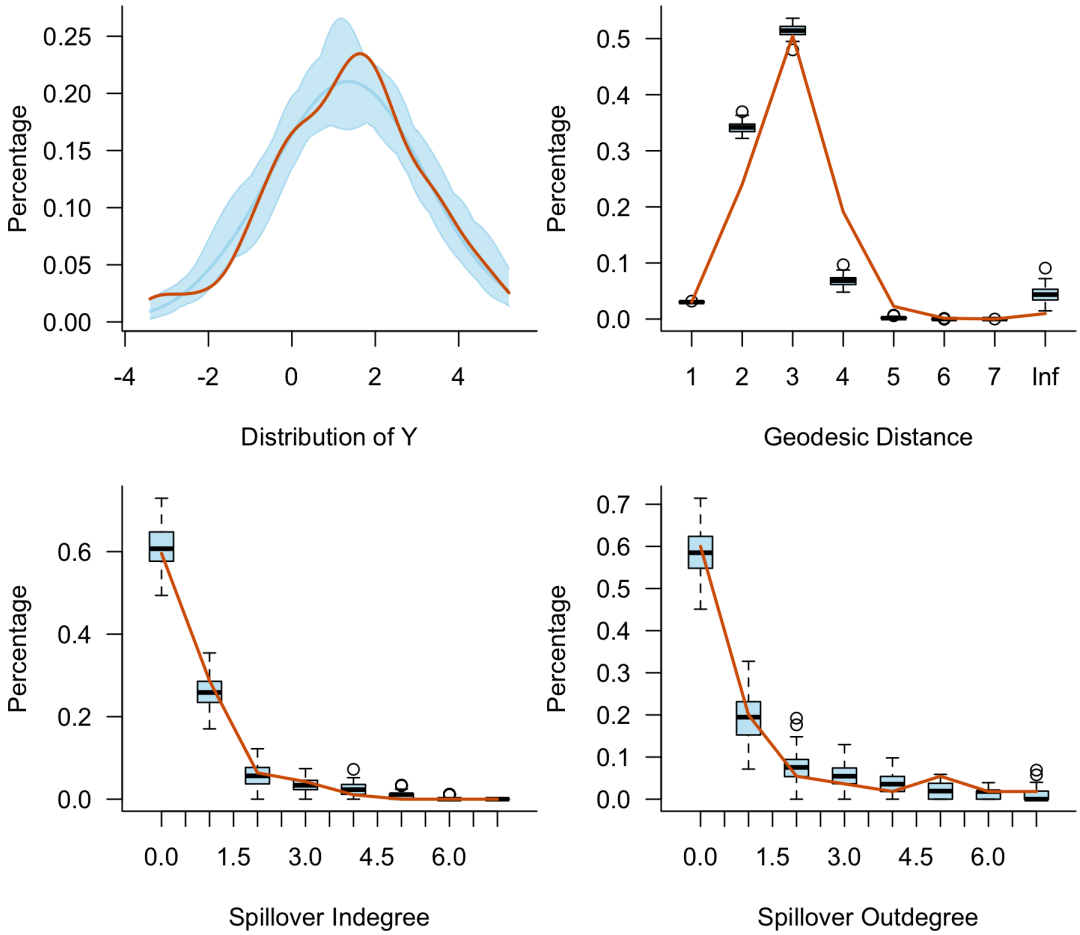

Figure 10: Copenhagen network study: model assessment in terms of the distribution of outcomes Y_i, geodesic distances, and spillover degrees.

Figure 10 suggests that the model matches the observed data in terms of the marginal distribution of outcomes Y_i, spillover in- and outdegrees, and geodesic distances.

6. Discussion

R package **iglm** provides a comprehensive regression framework for studying spillover and other phenomena in connected populations. That said, any question that can be raised about regression can be raised about regression under interference in connected populations. Many of these questions are open. We highlight two open questions.

First, we have assumed that the neighborhoods of units are known. While specifying the neighborhoods is possible in some applications—as demonstrated here, it may be infeasible in others. How to learn unknown neighborhoods from data is an interesting question that we intend to address in the future.

Second, R package **iglm** offers many possible model terms. In practice, users may wish to select a subset of model terms. Model selection with theoretical guarantees for dependent attributes (X_i, Y_i) and connections $Z_{i,j}$ is an open problem that we intend to address in future

releases of R package **iglm**.

Acknowledgments

We acknowledge support by U.S. National Science Foundation award NSF DMS-2515763 and U.S. Army Research Office award ARO W911NF-21-1-0335. We are grateful to Subhankar Bhadra and Donghwi Nam and for valuable feedback on drafts of this manuscript.

References

Brown L (1986). *Fundamentals of Statistical Exponential Families: With Applications in Statistical Decision Theory*. Institute of Mathematical Statistics, Hayworth, CA, USA.

Caimo A, Friel N (2014). "Bergm: Bayesian Exponential Random Graphs in R." *Journal of Statistical Software*, **61**, 1–25.

Chang W (2025). *R6: Encapsulated Classes with Reference Semantics*. doi:10.32614/CRAN.package.R6. R package version 2.6.1, URL https://CRAN.R-project.org/package=R6.

Csárdi G, Nepusz T, Traag V, Horvát S, Zanini F, Noom D, Müller K (2025). *igraph: Network Analysis and Visualization in R*. doi:10.5281/zenodo.7682609. R package version 2.1.4.

Eddelbuettel D (2013). *Seamless R and C++ Integration with Rcpp*. Springer, New York. doi:10.1007/978-1-4614-6868-4. ISBN 978-1-4614-6867-7.

Eddelbuettel D, Sanderson C (2014). "RcppArmadillo: Accelerating R with high-performance C++ linear algebra." *Computational Statistics and Data Analysis*, **71**, 1054–1063. doi:10.1016/j.csda.2013.02.005.

Fellows I, Clark D (2025). ***ernm**: Exponential-Family Random Network Models*. R package version 1.0.4, URL https://CRAN.R-project.org/package=ernm.

Fellows I, Handcock MS (2012). "Exponential-family Random Network Models." *Technical report*, Department of Statistics, University of California, Los Angeles. ArXiv:1208.0121.

Fosdick BK, Hoff PD (2015). "Testing and modeling dependencies between a network and nodal attributes." *Journal of the American Statistical Association*, **110**, 1047–1056.

Fritz C, Schweinberger M (2025). *iglm: Regression under Interference in Connected Populations*. R package version 1.2.4, URL https://CRAN.R-project.org/package=iglm.

Fritz C, Schweinberger M, Bhadra S, Hunter DR (2026). "A regression framework for studying relationships among attributes under network interference." *Journal of the American Statistical Association*. To appear.

Hoff P, Fosdick B, Volfovsky A (2024). *amen: Additive and Multiplicative Effects Models for Networks and Relational Data*. doi:10.32614/CRAN.package.amen. R package version 1.4.5, URL https://CRAN.R-project.org/package=amen.

Hoff PD (2021). "Additive and multiplicative effects network models." *Statistical Science*, **36**, 34–50.

Hunter DR, Handcock MS (2006). "Inference in curved exponential family models for networks." *Journal of Computational and Graphical Statistics*, **15**, 565–583.

Kim T, Nakka N, Gopal I, Desmarais BA, Mancinelli A, Harden JJ, Ko H, Boehmke FJ (2022). "Attention to the COVID-19 pandemic on Twitter: Partisan differences among U.S. state legislators." *Legislative Studies Quarterly*, **47**, 1023–1041. doi:10.1111/lsq.12367.

Koskinen J, Daraganova G (2022). "Bayesian analysis of social influence." *Journal of the Royal Statistical Society Series A: Statistics in Society*, **185**, 1855–1881.

Krivitsky PN, Hunter DR, Morris M, Klumb C (2023). "**ergm** 4: New Features for Analyzing Exponential-Family Random Graph Models." *Journal of Statistical Software*, **105**, 1–44. doi:10.18637/jss.v105.i06.

Meyer D, Zeileis A, Hornik K, Friendly M (2024). *vcd: Visualizing Categorical Data*. doi:10.32614/CRAN.package.vcd. R package version 1.4-13.

Niezink NMD, Snijders TAB (2017). "Co-evolution of social networks and continuous actor attributes." *The Annals of Applied Statistics*, **11**, 1948–1973.

Ortega JM, Rheinboldt WC (2000). *Iterative solution of nonlinear equations in several variables*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Plummer M, Best N, Cowles K, Vines K (2006). "CODA: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**, 7–11.

R Core Team (2026). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rinaldo A, Fienberg SE, Zhou Y (2009). "On the geometry of discrete exponential families with application to exponential random graph models." *Electronic Journal of Statistics*, **3**, 446–484.

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez JC, Müller M (2011). "pROC: an open-source package for R and S+ to analyze and compare ROC curves." *BMC Bioinformatics*, **12**, 77.

Sapiezynski P, Stopczynski A, Lassen DD, Lehmann S (2019). "Interaction Data from the Copenhagen Networks Study." *Scientific Data*, **6**, 315.

Snijders TAB, Ripley RM, Boitmanis K, Steglich C, Niezink NMD, Amati V, Schoenenberger F (2025). *Siena - Simulation Investigation for Empirical Network Analysis*. University of Groningen, Groningen, The Netherlands. R package version 1.5.0, URL https://www.stats.ox.ac.uk/~snijders/siena/.

Snijders TAB, Steglich CEG, Schweinberger M (2007). "Modeling the co-evolution of networks and behavior." In K van Montfort, H Oud, A Satorra (eds.), *Longitudinal models in the behavioral and related sciences*, pp. 41–71. Lawrence Erlbaum.

Wang P, Robins G, Pattison P, Koskinen J (2009). *PNet: program for the simulation and estimation of exponential random graph models*. Melbourne School of Psychological Sciences, The University of Melbourne.

Wang Z, Fellows IE, Handcock MS (2024). "Understanding networks with exponential-family random network models." *Social Networks*, **78**, 81–91.

A. Implemented Model Terms

Before stating all statistics, we introduce the following definitions:

Connections: Different type of indicators for connections:

- Overlapping: $u_{i,j} = c_{i,j} z_{i,j}$, a connection between units i and j where $\mathcal{N}_i \cap \mathcal{N}_j \neq \emptyset$.
- Non-overlapping: $k_{i,j} = (1 - c_{i,j}) z_{i,j}$, a connection between units i and j where $\mathcal{N}_i \cap \mathcal{N}_j = \emptyset$.
- $e_{i,j}^{(\mathtt{s})}$ for $\mathtt{s} \in \{\mathtt{global}, \mathtt{local}, \mathtt{alocal}\}$ is defined by:

$$e_{i,j}^{(\mathtt{s})} := \begin{cases} z_{i,j} & \text{, if } \mathtt{s} = \mathtt{global} \\ u_{i,j} & \text{, if } \mathtt{s} = \mathtt{local} \\ k_{i,j} & \text{, if } \mathtt{s} = \mathtt{alocal} \end{cases}$$

  The mode parameter `s` is generally defined as $\mathtt{s} \in \{\mathtt{global}, \mathtt{local}, \mathtt{alocal}\}$, but note that for the terms `gwesp`, `gwdsp`, `gwodegree`, `gwidegree`, `edges_x_match`, and `edges_y_match` (defined in Table 2), only the options $\mathtt{s} \in \{\mathtt{global}, \mathtt{local}\}$ are implemented as their `alocal` version is not very useful.

Degree Statistics: For unit $i \in \mathcal{P}$ and mode $s \in \{\mathtt{global}, \mathtt{local}\}$:

- Out-degree: $\deg(i, \mathtt{s}) = \sum_{j \in \mathcal{P} \setminus \{i\}} e_{i,j}^{(\mathtt{s})}$ with $\deg(i) = \deg(i, \mathtt{global})$.
- In-degree: $\mathrm{ideg}(i, \mathtt{s}) = \sum_{j \in \mathcal{P} \setminus \{i\}} e_{j,i}^{(\mathtt{s})}$ with $\mathrm{ideg}(i) = \mathrm{ideg}(i, \mathtt{global})$.

Common Partners (CP): For a dyad $(i, j) \in \mathcal{D}$ and mode $s \in \{\mathtt{global}, \mathtt{local}\}$, the number of shared partners via distinct path structures is defined as:

- Outgoing Two-Paths (OTP): $\mathrm{CP}(i, j, \mathtt{s}, \mathtt{OTP}) = \sum_{h \in \mathcal{P} \setminus \{i,j\}} e_{i,h}^{(\mathtt{s})} e_{h,j}^{(\mathtt{s})}$.
- Incoming Shared Partners (ISP): $\mathrm{CP}(i, j, \mathtt{s}, \mathtt{ISP}) = \sum_{h \in \mathcal{P} \setminus \{i,j\}} e_{h,i}^{(\mathtt{s})} e_{h,j}^{(\mathtt{s})}$.
- Outgoing Shared Partners (OSP): $\mathrm{CP}(i, j, \mathtt{s}, \mathtt{OSP}) = \sum_{h \in \mathcal{P} \setminus \{i,j\}} e_{i,h}^{(\mathtt{s})} e_{j,h}^{(\mathtt{s})}$.
- Incoming Two-Paths (ITP): $\mathrm{CP}(i, j, \mathtt{s}, \mathtt{ITP}) = \sum_{h \in \mathcal{P} \setminus \{i,j\}} e_{h,i}^{(\mathtt{s})} e_{j,h}^{(\mathtt{s})}$.
- Undirected Version: $\mathrm{CP}(i, j, \mathtt{s}) = \sum_{h \in \mathcal{P} \setminus \{i,j\}} e_{i,h}^{(\mathtt{s})} e_{h,j}^{(\mathtt{s})}$.

Miscellaneous:

- Geometrically-weighted weight: $w_k(\alpha) = \exp(\alpha) \left[1 - (1 - \exp(-\alpha))^k\right]$.
- Indicator for directionality: $\mathbb{I}_U(\boldsymbol{z})$, taking the value 1 if connections in $\boldsymbol{z}$ are undirected, and 0 otherwise.
- Indicator for transitive connection: $d_{i,j}(\boldsymbol{z}) = \mathbb{I}(\exists\, k \in \mathcal{N}_i \cap \mathcal{N}_j : z_{i,k} = z_{k,j} = 1)$

Table 2 is a lists all implemented terms as of **iglm** version 1.2.4 and will be extended in future releases.

Table 2: List of terms implemented in **iglm** to be included in functions $g_i(\cdot)$ and $h_{i,j}(\cdot)$. The first column states the name to be included on the right-hand side of the `formula` parameter. The second column gives the mathematical definitions of the $g_i(\cdot)$ and $h_{i,j}(\cdot)$ functions, respectively. The third column indicates whether the term is suitable for undirected connections (✔) or not (✖).

Name (Code)	**Term Definition**	**Undirected**
1. Attribute Dependence ($g_i(x_i, y_i)$ ***Terms)***		
`attribute_x`	x_i	✔
`attribute_y`	y_i	✔
`cov_x`	$v_i\, x_i$	✔
`cov_y`	$v_i\, y_i$	✔
`attribute_xy(mode = "global")`	$x_i\, y_i$	✔
`attribute_xy(mode = "local")`	$x_i \sum_{j\in\mathcal{N}_i} y_j + y_i \sum_{j\in\mathcal{N}_i} x_j$	✔
`attribute_xy(mode = "alocal")`	$x_i \sum_{j\notin\mathcal{N}_i} y_j + y_i \sum_{j\notin\mathcal{N}_i} x_j$	✔
2. Network Dependence ($h_{i,j}(x, y, z)$ ***Terms)***		
`degrees`	Degree Effects	✔
`edges(mode = "s")`	$e_{i,j}^{(\mathtt{s})}$	✔
`mutual(mode = "s")`	$e_{i,j}^{(\mathtt{s})}\, e_{j,i}^{(\mathtt{s})}/2$	✖
`cov_z(mode = "s")`	$w_{i,j} e_{i,j}^{(\mathtt{s})}$	✔
`cov_z_out(mode = "s")`	$v_i e_{i,j}^{(\mathtt{s})}$	✖
`cov_z_in(mode = "s")`	$v_j e_{i,j}^{(\mathtt{s})}$	✖
`isolates`	$\mathbb{I}(\sum_{j\in\mathcal{P}\setminus\{i\}} z_{i,j} + z_{j,i} = 0)$	✔
`nonisolates`	$\mathbb{I}(\sum_{j\in\mathcal{P}\setminus\{i\}} z_{i,j} + z_{j,i} \neq 0)$	✔
`gwdegree(mode = "global")`	$w_{\deg(i)}(\alpha) + w_{\deg(j)}(\alpha)$	✔
`gwodegree(mode = "s")`	$w_{\deg(i,\mathtt{s})}(\alpha)$	✖

Continued on next page

Table 2 – continued from previous page

Name (Code)	Term Definition	Undirected
`gwidegree(mode = "s")`	$w_{\text{ideg}(i,\mathtt{s})}(\alpha)$	✗
`transitive`	$d_{i,j}(\boldsymbol{z})\, z_{i,j}$	✓
`gwesp_symm(mode = "s")`	$e_{i,j}^{(\mathtt{s})}\, w_{\text{CP}(i,j,\mathtt{s})}(\alpha)$	✓
`gwesp(mode = "s", type = "[ITP/ISP/OTP/OSP]", decay = `α`)`	$e_{i,j}^{(\mathtt{s})}\, w_{\text{CP}(i,j,\mathtt{s},\mathtt{type})}(\alpha)$	✗
`gwdsp_symm(mode = "local")`	$w_{\text{CP}(i,j,\mathtt{local})}(\alpha)$	✓
`gwdsp(mode = "s", type = "[ITP/ISP/OTP/OSP]", decay = `α`)`	$w_{\text{CP}(i,j,\mathtt{s},\mathtt{type})}(\alpha)$	✗
3. Joint Attribute/Network Dependence ($h_{i,j}(x,y,z)$ ***Terms)***		
`attribute_xz(mode = "local")`	$(x_i + x_j)\, u_{i,j}$	✓
`attribute_yz(mode = "local")`	$(y_i + y_j)\, u_{i,j}$	✓
`edges_x_match(mode = "s")`	$\mathbb{I}(x_i = x_j)\, e_{i,j}^{(\mathtt{s})}$	✓
`edges_y_match(mode = "s")`	$\mathbb{I}(y_i = y_j)\, e_{i,j}^{(\mathtt{s})}$	✓
`outedges_x(mode = "s")`	$x_i\, e_{i,j}^{(\mathtt{s})}$	✗
`inedges_x(mode = "s")`	$x_j\, e_{i,j}^{(\mathtt{s})}$	✗
`outedges_y(mode = "s")`	$y_i\, e_{i,j}^{(\mathtt{s})}$	✗
`inedges_y(mode = "s")`	$y_j\, e_{i,j}^{(\mathtt{s})}$	✗
`spillover_xx(mode = "local")`	$x_i\, x_j\, u_{i,j}$	✓
`spillover_xx_scaled(mode = "s")`	$\left(\frac{x_i\, x_j}{\deg(i,\mathtt{s})} + \frac{x_j\, x_i}{\deg(j,\mathtt{s})}\mathbb{I}_U(\boldsymbol{z})\right) e_{i,j}^{(\mathtt{s})}$	✓
`spillover_yy(mode = "local")`	$y_i\, y_j\, u_{i,j}$	✓
`spillover_yy_scaled(mode = "s")`	$\left(\frac{y_i y_j}{\deg(i,\mathtt{s})} + \frac{y_j y_i}{\deg(j,\mathtt{s})}\mathbb{I}_U(\boldsymbol{z})\right) e_{i,j}^{(\mathtt{s})}$	✓
`spillover_xy(mode = "local")`	$x_i\, y_j\, u_{i,j} + x_j\, y_i\, u_{i,j}\, \mathbb{I}_U(\boldsymbol{z})$	✓
`spillover_xy_scaled(mode = "s")`	$\left(\frac{x_i\, y_j}{\deg(i,\mathtt{s})} + \frac{x_j\, y_i}{\deg(j,\mathtt{s})}\mathbb{I}_U(\boldsymbol{z})\right) e_{i,j}^{(\mathtt{s})}$	✓

Continued on next page

Table 2 – continued from previous page

Name (Code)	Term Definition	Undirected
`spillover_yx(mode = "local")`	$y_i \, x_j \, u_{i,j}$	✗
`spillover_yx_scaled(mode = "s")`	$\left(\frac{y_i \, x_j}{\deg(i, \mathtt{s})} + \frac{y_j \, x_i}{\deg(j, \mathtt{s})} \mathbb{I}_U(\boldsymbol{z}) \right) e_{i,j}^{(\mathtt{s})}$	✓
`spillover_yc(mode = "local")`	$c_{i,j}(v_j \, y_i + \mathbb{I}_U(\boldsymbol{z}) \, v_i \, y_j) z_{i,j}$	✓

B. Details on the C++ backend

The `XYZ_class` is the analog C++ class to the `iglm.data` R6 in R with the following main attributes:

- `int n_actor`: Number of units;
- `arma::vec x_attribute` and `arma::vec y_attribute`: Vector representations of $\boldsymbol{x}$ and $\mathbf{y}$;
- `std::vector<std::vector<int>> z_network.adj_list` and `std::vector<std::vector<int>> z_network.adj_list_in`: Representation of $\boldsymbol{z}$ via two edge lists: one for outgoing and one for ingoing connections. In case of undirected connections, `z_network.adj_list_in` is not defined;
- `std::vector<std::vector<int>> neighborhood`: Representation of $\mathcal{N}_i$ for $i = 1, ..., N$ via a neighborhood list;
- `std::vector<std::vector<int>> adj_list_nb` and `std::vector<std::vector<int>> adj_list_in_nb`: Two edge lists for connections with nonoverlapping neighborhoods: one for outgoing and one for ingoing connections. In case of undirected connections, `adj_list_in_nb` is not defined;

As shown in Section 4, the `XYZ_class` instance internally generated in package **iglm** is named `object`. Below is an exhaustive enumeration of the querying methods for this instance users can rely on to compute $\Delta_{\mathcal{X},i}, \Delta_{\mathcal{Y},i}$, and $\Delta_{\mathcal{Z},i,j}$. First, we list commands to evaluate specific parts of the data:

- **Attribute without scale:** To get the value x_i or y_i, call `object.x_attribute.get_val_no_scale(i)` or `object.y_attribute.get_val_no_scale(i)` for `int` i.
- **Attribute with scale:** To get the value x_i^* or y_i^*, call `object.x_attribute.get_val(i)` or `object.y_attribute.get_val(i)` for `int` i.
- **Connections:** To check if $z_{ij} = 1$, users must call `object.z_network.get_val(from, to)`.
- **Neighborhood Overlap:** To check if dyad (i, j) have overlapping neighborhoods, call `object.get_val_overlap(from, to)` for `int` i and `int` j.

Second, we list additional functions to ease define own C+- functions:

- **Out- and Ingoing Connections:** The `std::vector<int>` object of all connections from or to unit i can be assessed via `object.z_network.adj_list[i]` and `object.z_network.adj_list_in[i]`, respectively.
- **Out- and Indegree:** The out and indegree of unit i is given by `object.z_network.out_degrees.at(i)` and `object.z_network.in_degrees.at(i)`, respectively.

- **Out- and Indegree with Overlapping Neighborhood:** The out and indegree of unit i with other units that have some overlapping neighborhood is given by `object.out_degrees_nb.at(i)` and `object.in_degrees_nb.at(i)`, respectively.
- **Shared Partners:** To get the set of common partners k connecting i and j in the global network $\boldsymbol{z}$ as a `std::vector<int>` object, call `object.common_partners(from, to, type)`.
- **Shared Partners with Overlapping Neighborhood:** To get the set of common partners k connecting i and j such that all have overlapping neighborhoods as a `std::vector<int>` object, call `object.common_partners_nb(from, to, type)`.
- **Count Shared Partners:** To compute the number of intermediary nodes k connecting i and j in the global network $\boldsymbol{z}$, users invoke `object.count_common_partners(from, to, type)`. The `type` argument must specify the geometric path (e.g., "OTP" for Out-Two-Paths where $Z_{ik}\, Z_{kj} = 1$, see the `iglm` manual for full definitions). Here, the types of the arguments are `int` from, `int` to, and `string` type.
- **Count Shared Partners with Overlapping Neighborhood:** To compute for some pair of units the number of common partners that have an overlapping neighborhood, use `object.count_common_partners_nb(from, to, type)`.

Affiliation:

Cornelius Fritz
School of Computer Science and Statistics
Trinity College Dublin
O'Reilly Institute
Dublin 2, Republic of Ireland
E-mail: c.fritz@tcd.ie

Michael Schweinberger
Department of Statistics
The Pennsylvania State University
326 Thomas Building
University Park, PA 16802
E-mail: mus47@psu.edu