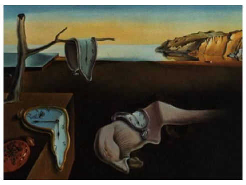


Journal of Statistical Software

February 2020, Volume 92, Issue 3. *doi: 10.18637/jss.v092.i03*

PAFit: An R Package for the Non-Parametric Estimation of Preferential Attachment and Node Fitness in Temporal Complex Networks

Thong Pham
RIKEN Center for AIP

Paul Sheridan
Hirosaki University

Hidetoshi Shimodaira
Kyoto University
RIKEN Center for AIP

Abstract

Many real-world systems are profitably described as complex networks that grow over time. Preferential attachment and node fitness are two simple growth mechanisms that not only explain certain structural properties commonly observed in real-world systems, but are also tied to a number of applications in modeling and inference. While there are statistical packages for estimating various parametric forms of the preferential attachment function, there is no such package implementing non-parametric estimation procedures. The non-parametric approach to the estimation of the preferential attachment function allows for comparatively finer-grained investigations of the "rich-get-richer" phenomenon that could lead to novel insights in the search to explain certain nonstandard structural properties observed in real-world networks. This paper introduces the R package **PAFit**, which implements non-parametric procedures for estimating the preferential attachment function and node fitnesses in a growing network, as well as a number of functions for generating complex networks from these two mechanisms. The main computational part of the package is implemented in C++ with **OpenMP** to ensure scalability to large-scale networks. In this paper, we first introduce the main functionalities of **PAFit** through simulated examples, and then use the package to analyze a collaboration network between scientists in the field of complex networks. The results indicate the joint presence of "rich-get-richer" and "fit-get-richer" phenomena in the collaboration network. The estimated attachment function is observed to be near-linear, which we interpret as meaning that the chance an author gets a new collaborator is proportional to their current number of collaborators. Furthermore, the estimated author fitnesses reveal a host of familiar faces from the complex networks community among the field's topmost fittest network scientists.

Keywords: temporal networks, dynamic networks, preferential attachment, fitness, rich-get-richer, fit-get-richer, R, C++, **Rcpp**, **OpenMP**.

1. Introduction

Since the end of the last century, complex networks have been increasingly used in modeling many temporal relations found in diverse fields (Dorogovtsev and Mendes 2003; Caldarelli 2007; Newman 2010). Some notable examples include collaboration networks between authors in a scientific field (Newman 2001), connection networks between computers on the Internet (Barabási, Albert, and Jeong 2000), and sexual relation networks between members of a community (Liljeros, Edling, Amaral, Stanley, and Aberg 2001). The primary motivation for using complex networks as a simplified representation of real-world systems is that they shed light on the behaviors of complex systems through the study of underlying patterns of connections. Although this is an over-simplification for systems depending heavily on domain-specific details, this approach nevertheless offers a first view of a system's topological properties, and can be used to guide subsequent in-depth analyses.

Among the most important real-world network structural properties is degree distribution. Degree distribution lets us understand the proportion of highly and lowly connected nodes in a network. Since the highly connected nodes are key components of a network, this understanding in turn sheds light on the answers of important practical questions, including how to prevent the spreading of rumors (Nekovee, Moreno, Bianconi, and Marsili 2007), how to stop a virus outbreak (Pastor-Satorras and Vespignani 2001), and how to guard against cybernetic attacks (Albert, Jeong, and Barabási 2000).

The degree distributions of many real-world networks have been found to be heavy-tailed (Albert and Barabási 1999). The best-known heavy-tailed distribution in network science is the power-law, which is a distribution where the number of nodes in a network with degree k is proportional to $k^{-\gamma}$ with $\gamma > 1$. Besides the power-law, there is emerging evidence that real-world network degree distributions have other heavy-tailed forms, including the log-normal (Redner 2005), exponential (Dunne, Williams, and Martinez 2002), stretched exponential (Newman, Forrest, and Balthrop 2002), and power-law with exponential cut-off (Clauset, Shalizi, and Newman 2009).

All of these heavy-tailed distributions differ from the light-tailed binomial degree distribution, which is characteristic of networks produced by the classical Erdös-Rényi (ER) random graph model (Erdös and Rényi 1959). This observation prompted network scientists to search for new modeling ingredients capable of explaining heavy-tailed degree distributions. What they found is that temporal complex network models incorporating growth mechanisms offer a powerful modeling framework for achieving this end.

Temporal complex network models, or temporal network models for short, are probabilistic generative models of real-world networks that change with time. In its most common form, a temporal network model assumes that a network grows gradually from some initial state by the addition of new nodes and edges over a large number of discrete time-steps. Some well-known basic models in the field of complex networks are the Barabási-Albert (BA) model (Albert and Barabási 1999) and the Bianconi-Barabási (BB) model (Bianconi and Barabási 2001). More complex growth models that are used in the field include exponential random graph models (Ripley, Snijders, Bóda, Vörös, and Preciado 2018; Krivitsky and Handcock 2019) and dynamic stochastic block models (Matias and Miele 2016). Growth mechanisms, which govern how a node acquires new edges in the growth process, are the most important elements that distinguish different temporal network models.

This paper focuses on estimating two interpretable growth mechanisms: preferential attachment (PA) and node fitness. In the PA mechanism, the probability $P_i(t)$ that a node v_i acquires a new edge at time t is proportional to a positive function, $A_{k_i(t)}$, of its current degree $k_i(t)$. The function A_k is called the attachment function. The name "preferential attachment" stems from the motivation for the mechanism: If A_k is an increasing function, a highly connected node will acquire more edges than a lowly connected node, which is an appealing property in many real-world situations. From now on, we will say that PA exists if A_k is an increasing function. The opposite of PA, which we call anti-PA, occurs when A_k is a decreasing function. Note, however, that the meaning we use here differs from the original meaning of the term "preferential attachment" used in the BA model, which means only the linear case of $A_k = k$. This linear form in fact has been long known in other fields with various names such as "rich-get-richer" (Simon 1955) and "cumulative advantage" (de Solla Price 1976). When A_k assumes the log-linear form of k^α, with α called the attachment exponent, we have the generalized BA model (Krapivsky, Rodgers, and Redner 2001).

In contrast with the PA mechanism, in the fitness mechanism the probability $P_i(t)$ that a node v_i acquires a new edge depends only on the positive number η_i. The quantity η_i is called the node fitness, or just fitness, of v_i and can be interpreted as its intrinsic attractiveness. The fitness mechanism offers a simple way to express the variance in edge-acquisition potential between nodes of the same degree. For example, two early-career scientists with roughly the same number of collaborators at some point in time may acquire different numbers of collaborators in the future based on their intrinsic fitnesses.

The PA and node fitness mechanisms combine to produce a wide range of degree distributions. In their combined form, the probability $P_i(t)$ is proportional to the product of $A_{k_i(t)}$ and η_i:

$$P_i(t) \propto A_{k_i(t)} \times \eta_i. \tag{1}$$

Based on the functional form of A_k and the distribution of η_i's, the model defined by Equation 1 can produce networks with various degree distributions (Bianconi and Barabási 2001; Caldarelli, Capocci, De Los Rios, and Muñoz 2002; Borgs, Chayes, Daskalakis, and Roch 2007; Kong, Sarshar, and Roychowdhury 2008). In Section 2, we will discuss the relation of Equation 1 with existing statistical models.

Equation 1 has a number of applications. Based on the functional forms of A_k and η_i, we can test for the presence of one and/or the other of the "rich-get-richer" and "fit-get-richer" phenomena in a temporal network (Pham, Sheridan, and Shimodaira 2016). These two mechanisms have been advanced to explain another phenomenon called the "generalized friendship paradox" (Feld 1991; Eom and Jo 2014; Momeni and Rabbat 2015). They are also used in inference problems in biological networks (Sheridan, Kamimura, and Shimodaira 2010; Guetz and Holmes 2011), the World Wide Web (Kong *et al.* 2008), Internet topology graphs (Bezáková, Kalai, and Santhanam 2006), and citation networks (Wang, Song, and Barabási 2013; Sinatra, Wang, Deville, Song, and Barabási 2016; Ronda-Pupo and Pham 2018). Finally, we can classify real-world temporal network data based on the estimated attachment exponent of A_k (Kunegis, Blattner, and Moser 2013).

While there are existing R (R Core Team 2019) packages that estimate PA in a growing network, including the packages **tergm** (Krivitsky and Handcock 2019) and **RSiena** (Ripley *et al.* 2018), these packages, however, rely on parametric methods to estimate the A_k function. This means that one has to assume a functional form for A_k, rather than learning it from observed data without constraint. Non-parametric estimation of A_k allows for a finer inspection of the "rich-get-richer" phenomenon (Pham, Sheridan, and Shimodaira 2015; Pham *et al.* 2016), and such methods have been used to provide clues to explain irregularities observed in real-world degree distributions (Sheridan and Onodera 2018).

This paper introduces the R package **PAFit** (Pham, Sheridan, and Shimodaira 2020), which is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=PAFit`. The package fills the gap with an implementation of the standard PA and node fitness non-parametric estimation procedures. In particular, we implement Jeong's method (Jeong, Néda, and Barabási 2003), Newman's method (Newman 2001) and the PAFit method (Pham *et al.* 2015, 2016) in the package. The first two are heuristic methods that are widely used to estimate non-parametrically the attachment function A_k in isolation, while the last one is a statistical method that can non-parametrically estimate either A_k (or η_i) in isolation or simultaneously estimate the two mechanisms. Although using PAFit is advisable in almost every circumstance, Jeong's method and Newman's method are still widely used and might still be appropriate in certain situations. Therefore, the inclusion of the two heuristic methods in the package is warranted. We discuss their strengths and shortcomings in Section 2 when we provide an overview of the methodology and related statistical models.

The package also implements a variety of functions to simulate temporal networks from the PA and node fitness mechanisms, as well as functions to plot the estimated results and underlying uncertainties. We review **PAFit**'s main functions in Section 3. Before demonstrating their usages on three simulated examples in Section 5, we discuss how **PAFit** relates with existing network analysis packages in Section 4. We provide a systematic simulation to assess the results of the non-parametric joint estimation in Section 6, before showing a complete end-to-end work-flow analyzing a collaboration network of scientists from the field of complex networks in Section 7. Finally, concluding remarks are given in Section 8.

2. Mathematical background

Here we review the standard methods for estimating the attachment function A_k and node fitnesses η_i in a temporal network. In Section 2.1, we state the network growth model used in the package as well as discuss its relation with exiting statistical models. We review the estimation of A_k in isolation in Section 2.2, then the estimation of the η_i's in isolation in Section 2.3, and finally the joint estimation of A_k and the η_i's in Section 2.4.

2.1. Network model

First we describe the general temporal (GT) model (Pham *et al.* 2016) used in **PAFit**. The model is a generalization of many well-known temporal network models in the complex network field.

Starting from some given initial graph G_0, the GT model generates a temporal network sequentially as follows: At time-step $t \geq 1$, the network G_t is obtained by adding new edges and new nodes to G_{t-1}. The number of new edges and new nodes added at time t is denoted as $m(t)$ and $n(t)$, respectively. The model assumes that the parameters governing the distributions of G_0, $m(t)$ and $n(t)$ do not involve A_k and the η_i's. Note that the term $k_i(t)$ is defined as the degree of node v_i at the onset of time-step t. On top of these structural preconditions, the GT model assumes that the probability that a node v_i with degree $k_i(t)$ receives a new edge at time t is given by the formula of Equation 1.

The GT model includes a handful of well-known growing network models, based on PA and

Temporal network model	Attachment function	Node fitness
Growing ER model (Callaway *et al.* 2001)	$A_k = 1$	$\eta_i = 1$
BA model	$A_k = k$	$\eta_i = 1$
Caldarelli model	$A_k = 1$	Free
BB model	$A_k = k$	Free

Table 1: Some well-known special cases of the GT model as defined by Equation 1.

node fitness as special cases; see Table 1 for a summary. Unlike the BA or BB models, the GT model allows for the emergence of new edges between old nodes and can handle both undirect and directed networks. We refer readers to Supplementary Information Section S2.2 in Pham *et al.* (2016) for the definition of the model in the case of undirected networks. The GT model is related to models used in the R packages **RSiena** and **tergm**. But we defer a discussion of these packages to Section 4.

Looking beyond the field of complex networks, the GT model bears some similarities to the contagious Poisson process (Coleman 1964; Allison 1980) and the conditional frailty model (Kelly and Lim 2000; Box-Steffensmeier and De Boef 2006). In the contagious Poisson process, the initial propensity of each node plays a similar role to that of node fitness and the rate of enforcement represents the PA mechanism. In the conditional frailty model, where the frailty of each node describes the heterogeneity among nodes and thus is similar to node fitness, the event-based baseline hazard rate has the same effect as the non-parametric function A_k.

2.2. Attachment function estimation

The methods for estimating the attachment function A_k in isolation assume a simplified version of Equation 1, in which the η_i are assumed to be 1. Thus the probability $P_i(t)$ in Equation 1 depends only on A_k. Perhaps the most frequently-encountered parametric version of this model is the log-linear form $A_k = k^\alpha$ with attachment exponent α. Network scientists are particularly interested in estimating α, since the asymptotic degree distribution of the network corresponds to simple regions of α. If α is less than unity (the sub-linear case), then the degree distribution is a stretched exponential, while in the super-linear case of $\alpha > 1$, one node will eventually get all the incoming new edges (Krapivsky *et al.* 2001). It is only the linear case of $\alpha = 1$ that gives rise to a power-law distribution.

Concerning the above model, there are three main methods for estimating A_k: Jeong's method (Jeong *et al.* 2003), Newman's method (Newman 2001), and PAFit (Pham *et al.* 2015). Jeong's method basically makes a histogram of the number of new edges n_k connected to a node with degree k, then divides n_k by the number of nodes with degree k in the system to get A_k (Jeong *et al.* 2003). Jeong's method has the merit of being simple, but estimates obtained using the method are subject to high variance and low accuracy (Pham *et al.* 2015). By contrast, Newman's method combines a series of histograms for lower variance and higher accuracy (Newman 2001). Note that in **PAFit** we implemented a corrected version of Newman's original method (Pham *et al.* 2015). The main drawback of Newman's method is that the mathematical assumption behind its derivation holds only when $\alpha = 1$, thus the method amounts to an approximation when $\alpha \neq 1$ (Pham *et al.* 2015).

The final method is PAFit (Pham *et al.* 2015). It iteratively maximizes an objective function that is a combination of the log-likelihood of the model with a regularization term for A_k by a minorization-maximization (MM) algorithm (Hunter and Lange 2000). There is a hyper-parameter, called r, in the method that controls the strength of the regularization. PAFit chooses r automatically by cross-validation (Pham *et al.* 2016). We defer the details to Section 2.4. The method is not only able to recover A_k accurately, but also can estimate the standard deviation of the estimated A_k for each k (Pham *et al.* 2015). Its main drawback is that it might be slow, since it is an iterative algorithm.

2.3. Node fitness estimation

When we consider only node fitnesses, there are two generative models in the literature with different assumptions regarding the functional form of A_k in Equation 1. While the Caldarelli model (Caldarelli *et al.* 2002) assumes that A_k is 1 for all k, the BB model (Bianconi and Barabási 2001) assumes that $A_k = k$. Both models have been shown to generate networks with various heavy-tailed distributions (Borgs *et al.* 2007; Kong *et al.* 2008).

Node fitnesses in both models can be estimated by variants of the PAFit method proposed in Pham *et al.* (2016), by either setting $A_k = k$ for the BB model or $A_k = 1$ for the Caldarelli model. These estimation methods use MM algorithms that maximize the corresponding log-likelihood functions with a regularization term that regularizes the distribution of the η_i's. More specifically, the inverse variance of this distribution is controlled by a hyper-parameter, called s, which is chosen automatically by cross-validation. We defer a more detail discussion to the next section. We note that node fitnesses in the BB model can also be estimated by the method in Kong *et al.* (2008). But since PAFit has been shown to outperform this method (Pham *et al.* 2016), we did not include it in the package.

2.4. Joint estimation of the attachment function and node fitnesses

Finally, by using the full model in Equation 1 the method PAFit in Pham *et al.* (2016) can jointly estimate A_k and η_i. We note that this full model includes all the temporal network models shown in Table 1. For a more complete table, see Table 1 in Pham *et al.* (2016).

The objective function of PAFit is a combination of the log-likelihood of the full model defined by Equation 1 and two regularization terms: one for A_k and one for η_i. While we refer readers to Supplementary Information Section S2.3 in Pham *et al.* (2016) for a complete presentation, we will sketch here the log-likelihood function for the case of directed networks. Assume the set of observed snapshots is $\{G_t\}_{t=0}^{T}$. Let $\mathbf{A} = [A_0\ A_1 \cdots A_{K-1}]^\top$ be the vector of the PA function and $\boldsymbol{\eta} = [\eta_1\ \eta_2 \cdots \eta_N]$ be the vector of node fitnesses. Here K is the maximum degree appearing in the growth process and N is the total number of nodes at the end of the process. Let $z_i(t)$ be the number of new edges connected to node v_i at time-step t. Equation 1 implies that $\{z_i(t)\}_{i=1}^{N}$ follows a multinomial distribution with parameters $\{\pi_i(t)\}_{i=1}^{N}$, where

$$\pi_i(t) = \frac{A_{k_i(t)}\eta_i}{\sum_{j=1}^{N} A_{k_j(t)}\eta_j}. \tag{2}$$

Here we use the convention $k_j(t) = -1$ for a node that did not exist at time-step t and $A_{-1} = 0$. Using Equation 2, one can write the likelihood of each snapshot $G_1, \cdots, G_T$. The log-likelihood function of the temporal network $\{G_t\}_{t=0}^{T}$ is then the sum of the log-likelihood of each snapshot and is equivalent to:

$$l(\mathbf{A}, \boldsymbol{\eta}) = \sum_{t=1}^{T}\sum_{i=1}^{N} z_i(t) \log A_{k_i(t)} + \sum_{t=1}^{T}\sum_{i=1}^{N} z_i(t) \log \eta_i - \sum_{t=1}^{T}\sum_{i=1}^{N} z_i(t) \log \sum_{j=1}^{N} A_{k_j(t)}\eta_j + C, \quad (3)$$

with C being the logarithm of the product of the probability mass functions of G_0, $m(t)$, and $n(t)$. Since the GT model, as stated in Section 2.1, assumes that the parameters governing the distributions of G_0, $m(t)$ and $n(t)$ do not involve A_k and η_i, we can treat C as a constant.

The regularization term for A_k is defined by

$$\mathit{reg}_A = -r \sum_{k=1}^{K-2} w_k \left(\log A_{k+1} + \log A_{k-1} - 2 \log A_k\right)^2, \quad (4)$$

with $r \geq 0$, $w_k = \sum_{t=1}^{T} m_k(t)$ and $m_k(t)$ the number of edges that connect to a degree k node at time-step t. This term controls the shape of A_k. When r is large, A_k becomes more linear on a log-scale. In limiting as r approaches infinity, we effectively assume that $A_k = k^\alpha$ (Pham *et al.* 2016). Thus this covers the case of $\alpha = 1$ in the BA model and the BB model, and the case of $\alpha = 0$ in the growing ER and the Caldarelli model.

The regularization term for the node fitnesses is defined by

$$\mathit{reg}_F = \sum_{i=1}^{N}((s-1)\log \eta_i - s\eta_i), \quad (5)$$

with $s > 0$. This term is the sum of the logarithms of Gamma distribution densities with mean 1 and variance $1/s$. The regularization is equivalent to placing such Gamma distributions as priors independently for each node fitness η_i. The larger the value of s, the more tightly concentrated the values of η_i become. If s is infinitely large, then all η_i will take the same value. This is equivalent to estimating the attachment function in isolation.

To conclude, joint estimation with the above regularization terms also compasses the two cases of estimating either A_k or η_i in isolation. In particular, we maximize the following objective function:

$$J(\mathbf{A}, \boldsymbol{\eta}) = l(\mathbf{A}, \boldsymbol{\eta}) + \mathit{reg}_A + \mathit{reg}_F,$$

with an MM algorithm. At each iteration, the algorithm replaces the objective function with an easier-to-maximize surrogate function and this surrogate function is maximized instead. The surrogate function is chosen in such a way that the objective function value is guaranteed to be nondecreasing over iterations. We refer the readers to Hunter and Lange (2004) for the definition of a surrogate function and the techniques used to derive them. For a surrogate function, the variables are often separable, i.e., the partial derivative of one variable does not involve the others, and thus the maximization at each iteration, i.e., finding the variables by setting all the partial derivatives to zero, can be parallelized. While we refer readers to Supplementary Information Section S2.4 of Pham *et al.* (2016) for a detailed discussion, the essence of the MM algorithms in **PAFit** is to linearize the term $\log \sum_{j=1}^{N} A_{k_j(t)}\eta_j$ in Equation 3 and to apply Jensen's inequality to make the variables in Equation 4 separable.

As mentioned in the two previous sections, the values of r and s are automatically selected by cross-validation. In particular, the dataset is divided into a learning part $\{G_t\}_0^{T_*}$ and a testing part $\{G_t\}_{T_*}^{T}$, where T_* is the smallest positive number such that the ratio of the number of new edges in the learning part, i.e., $\sum_{t=1}^{T_*}\sum_{i=1}^{N} z_i(t)$, to that of the whole dataset, i.e., $\sum_{t=1}^{T}\sum_{i=1}^{N} z_i(t)$, is at least $p = 0.75$ (the default value). For each combination of r and s, we use the learning data to get the estimated value of $\mathbf{A}$ and $\boldsymbol{\eta}$ and plug these estimated values into Equation 3 to calculate the log-likelihood of the testing data. The combination of r and s that maximize this log-likelihood is then chosen. The method then re-estimates $\mathbf{A}$ and $\boldsymbol{\eta}$ using the whole dataset with the chosen combination of r and s.

3. Package overview

The **PAFit** package provides functions to simulate various temporal network models, gather essential network statistics from raw input data, and use these summarized statistics in the estimation of A_k and the η_i values. The heavy computational parts of the package are implemented in `C++` through the use of the **Rcpp** package (Eddelbuettel and François 2011; Eddelbuettel 2013; Eddelbuettel and Balamuta 2017). Furthermore, multi-core machine users can enjoy a hassle-free speed up through **OpenMP** parallelization mechanisms implemented in the code. Apart from the main functions, the package also includes a real-world collaboration network dataset between scientists in the field of complex networks. Table 2 summarizes the main functions in the package. In what follows, we will review the main package functions one by one.

Firstly, most well-known temporal network models based on PA and node fitness mechanisms can be easily simulated using the package. **PAFit** implements `generate_BA` for the BA model, `generate_ER` for the growing ER model, `generate_BB` for the BB model, and `generate_fit_only` for the Caldarelli model. These functions have many customizable options. For example, the number of new edges at each time-step is a tunable stochastic variable; see Table 3 for descriptions of the parameters. They are actually wrappers of the more powerful `generate_net` function, which simulates networks with more flexible attachment function and node fitness settings.

Each temporal network model generation function outputs a '`PAFit_net`' object, which is a list with four fields: `type`, `fitness`, `PA`, and `graph`. The `type` field is a string indicating the type of network: `"directed"` or `"undirected"`. This field is `"directed"` for the networks generated by the simulation functions. The `fitness` and `PA` fields contain the true node fitnesses and PA function, respectively. The `graph` field contains the generated temporal network in a three-column matrix format. Each row of this matrix is of the form `(id_1 id_2 time_stamp)`. While `id_1` and `id_2` are IDs of the source node and the destination node, respectively, `time_stamp` is the birth time of the edge. This is the so-called edge-list format in which raw temporal networks are stored in many on-line repositories (Kunegis 2013; Leskovec and Krevl 2014). We will discuss how to use functions provided by **PAFit** to convert this edge-list format to formats used in other network analysis packages in the next section. One can apply the function `plot` directly to a '`PAFit_net`' object to visualize its contents.

The second functionality of **PAFit** is implemented in `get_statistics`. With its core part implemented in `C++`, this function efficiently collects all temporal network summary statistics that are needed in the subsequent estimation of PA and node fitnesses. The input network is assumed to be stored as a '`PAFit_net`' object. One can use the function `graph_from_file` to read an edge-list graph from a text file into a '`PAFit_net`' object, or convert an edge-list matrix to this class by the function `as.PAFit_net`.

Function	Main input	Output
`generate_ER`	Network parameters	Network from the growing ER model
`generate_BA`	Network parameters	Network from the generalized directed BA model
`generate_BB`	Network parameters	Network from the BB model
`generate_fit_only`	Network parameters	Network from the Caldarelli model
`generate_net`	Network parameters	Network from the GT model
`get_statistics`	‘`PAFit_net`’ object containing the network	‘`PAFit_data`’ object containing summary statistics
`Jeong`	‘`PAFit_net`’ object and ‘`PAFit_data`’ object	Estimated PA function by Jeong’s method
`Newman`	‘`PAFit_net`’ object and ‘`PAFit_data`’ object	Estimated PA function by Newman’s method
`only_A_estimate`	‘`PAFit_net`’ object and ‘`PAFit_data`’ object	Estimated PA function by PAFit
`only_F_estimate`	‘`PAFit_net`’ object and ‘`PAFit_data`’ object	Estimated node fitnesses by PAFit
`joint_estimate`	‘`PAFit_net`’ object and ‘`PAFit_data`’ object	Estimated PA function and node fitnesses by PAFit
`to_networkDynamic`	‘`PAFit_net`’ object	‘`networkDynamic`’ object
`from_networkDynamic`	‘`networkDynamic`’ object	‘`PAFit_net`’ object
`to_igraph`	‘`PAFit_net`’ object	‘`igraph`’ object
`from_igraph`	‘`igraph`’ object	‘`PAFit_net`’ object
`graph_to_file`	‘`PAFit_net`’ object	Text file in either edge-list format or `gml`
`graph_from_file`	Text file in edge-list format or `gml`	A ‘`PAFit_net`’ object
`as.PAFit_net`	Edge-list matrix	‘`PAFit_net`’ object
`test_linear_PA`	Degree vector	‘`Linear_PA_test_result`’ object

Table 2: Summary of the main functions in the **PAFit** package.

Parameter (default)	Description
`N` (1000)	Total number of nodes in the network
`num_seed` (2)	Initial graph is a circle with `num_seed` nodes
`multiple_node` (1)	Number of new nodes added at each time-step
`m` (1)	Number of edges of a new node
`alpha` (1)	Attachment exponent α when we assume $A_k = k^\alpha$
`mode_f` (`"gamma"`)	Distribution of node fitnesses: gamma, log-normal or power-law
`s` (10)	Distribution of node fitnesses has mean 1 and variance $1/s$

Table 3: Main parameters in network generating functions in the **PAFit** package.

The edge-list matrix is assumed to be in the same format as **PAFit** simulated graphs, i.e., each row is of the form `(id_1 id_2 time_stamp)`. The node IDs are required to be integers greater than -1, but need not to be contiguous. Note that `(id -1 t)` describes a node `id` that appeared at time `t` without any edge. There are no assumptions on the values or data types of `time_stamp`, other than that their chronological order is the same as what the R function `order` returns. Examples of timestamps that satisfy this requirement are the integer vector `1:T`, the format "yyyy-mm-dd", and the POSIX time.

The `get_statistics` function automatically handles both directed and undirected networks. It returns a list containing many statistics that can be used to characterize the network growth process. Notable fields are `m_tk` containing the number of new edges that connect to a degree-k node at time-step t, and `node_degree` containing the degree sequence, i.e., the degree of each node at each time-step.

The most important functionality of **PAFit** relates to the estimation of the attachment function and node fitnesses of a temporal network. This is implemented through various methods. There are three usages: estimation of the attachment function in isolation, estimation of node fitnesses in isolation, and the joint estimation of the attachment function and node fitnesses.

The functions for estimating the attachment function in isolation are: `Jeong` for Jeong's method, `Newman` for Newman's method, and `only_A_estimate` for the PAFit method in Pham *et al.* (2015). For estimation of node fitnesses in isolation, `only_F_estimate` implements a variant of the PAFit method in Pham *et al.* (2016). For the joint estimation of the attachment function and node fitnesses, we implement the full version of the PAFit method (Pham *et al.* 2016) in `joint_estimate`. The input of these functions is the output object of the function `get_statistics`. The output objects of these functions contain the estimation results as well as some additional information pertaining to the estimation process.

In Table 4, we show the input parameters of `joint_estimate`, the most important function in **PAFit**. This function takes the temporal network `net_object` and the summarized statistics `net_stat` as the main inputs. There are three parameters that control the estimation process: `p`, `stop_cond`, and `mode_reg_A`. The parameter `p` specifies the ratio of the number of new edges in the learning data to that of the full data in the cross-validation step. Following Pham *et al.* (2016), its default value is set at 0.75. The parameter `stop_cond` specifies the threshold ϵ: the iterative algorithm will continue until the relative difference in the objective function $J(\mathbf{A}, \boldsymbol{\eta})$ between two successive iterations falls below this threshold (Pham *et al.* 2016; Zhou, Alexander, and Lange 2011). The default value $\epsilon = 10^{-8}$ is set following Pham *et al.* (2016). The parameter `mode_reg_A` specifies the regularization term for A_k. The de-

Parameter	Default value
`net_object`	No default value
`net_stat`	`get_statistics(net_object)`
`p`	0.75
`stop_cond`	10^{-8}
`mode_reg_A`	0

Table 4: Parameters of the `joint_estimate` function and their default values.

fault value `mode_reg_A = 0` corresponds to the regularization term in Equation 4 (Pham *et al.* 2016). When `mode_reg_A = 1`, the following regularization term is used:

$$-r \sum_{k=2}^{K-1} w_k \left\{ \frac{\log A_{k+1} - \log A_k}{\log(k+1) - \log k} - \frac{\log A_k - \log A_{k-1}}{\log k - \log(k-1)} \right\}^2 . \tag{6}$$

Although this regularization term will enforce exactly the form $A_k = k^\alpha$, it is significantly slower to optimize this regularization term while the improvement over Equation 4 is little.

Finally, although one can roughly assess whether PA exists in the network by visual inspection of the estimated PA function, Handcock and Jones (2004) provide a method to test whether the linear PA-only case, i.e., $A_k = k$ and $\eta_i = 1$, is consistent with a given degree vector. We implemented this method in the function `test_linear_PA`. This function chooses the best fitted distribution to a given degree vector among a set of distributions by comparing the Akaike information criterion (AIC; Akaike 1974) or the Bayesian information criterion (BIC; Raftery 1995). The set of distributions are Yule, Waring, Poisson, geometric, and negative binomial. The linear PA-only case corresponds to Yule or Waring (Yule 1925; Irwin 1963).

4. Related network packages

Since network analysis has been an important field for a long time, various aspects of it have been implemented in a large number of software packages. To our best effort, we have confirmed that the non-parametric joint estimation of PA and fitness mechanisms in a growing network is not implemented elsewhere. Restricting the discussion to packages in R, there are some notable implementations of related statistical network models. For example, stochastic block models in the packages **igraph** (Csardi and Nepusz 2006), **sna** (Butts 2019), **blockmodels** (INRA and Leger 2015) and **dynsbm** (Matias and Miele 2016, 2019); exponential random graph models in the packages **ergm** (Hunter, Handcock, Butts, Goodreau, and Morris 2008; Handcock, Hunter, Butts, Goodreau, Krivitsky, and Morris 2018), **tergm** (Krivitsky and Handcock 2019), **hergm** (Schweinberger and Luna 2018), **btergm** (Leifeld, Cranmer, and Desmarais 2018), and **RSiena** (Ripley *et al.* 2018); and latent space models in the package **latentnet** (Krivitsky and Handcock 2008, 2018).

The **dynsbm** package estimates a dynamic stochastic block model in which nodes are assumed to belong to some latent groups which can vary with time, and the edge weight between two nodes at any time follows some parametric distribution. The package can deal with both discrete and continuously weighted edges.

The **igraph** package contains the functions `sample_pa` and `sample_growing` which are the equivalents of `generate_BA` and `generate_ER` in **PAFit**, respectively. Although **igraph** also generates networks from many other mechanisms, it does not contain any function for estimating the PA function and/or node fitnesses. It does contain many functionalities for dealing with stochastic block models and various other network models.

Some of the above packages are included in the extensive meta-package **statnet** (Handcock, Hunter, Butts, Goodreau, and Morris 2008; Handcock *et al.* 2019). In **statnet**, packages that deal with temporal networks are: **networkDynamic** (Butts, Leslie-Cook, Krivitsky, and Bender-deMoll 2019), **tsna** (Bender-deMoll and Morris 2019), and **tergm**. The **networkDynamic** package provides the '`networkDynamic`' class to store dynamic networks and various functions to manipulate them. The **tsna** package calculates many temporal statistics of a dynamic network stored in a '`networkDynamic`' object.

The closest packages to **PAFit** that estimate PA in a temporal network are **tergm** and **RSiena**, which implement sophisticated continuous-time and discrete-time Markov models. Regarding PA, all the implemented options in **tergm** and **RSiena** pertain to the parametric estimation of A_k, in contrast to the non-parametric estimation methods implemented in **PAFit**. Although it might be theoretically possible to describe a non-parametric A_k function in **tergm** and **RSiena**, they contain no regularization terms for the joint estimation of the non-parametric PA function and node fitnesses. Joint estimation without regularization terms is very likely unable to recover the true parameters, since the number of parameters is typically high. On the other hand, **PAFit** is specifically designed for estimating A_k non-parametrically with node fitnesses, since it has two regularization terms in Equations 4 and 5, together with the cross-validation step for selecting suitable regularization parameters.

PAFit provides functionalities to communicate with existing network analysis packages. Using `to_networkDynamic` and `from_networkDynamic`, one can convert a '`PAFit_net`' object to a **networkDynamic**'s '`networkDynamic`' object and vice versa. The functions `to_igraph` and `from_igraph` do the same for **igraph**'s '`igraph`' objects. The extensive functions of **statnet** and **igraph** packages can then be used. One can also output the graph stored in a '`PAFit_net`' object to the universal `gml` format by the function `graph_to_file`, or read from a `gml` file by the function `graph_from_file`.

5. Package usage

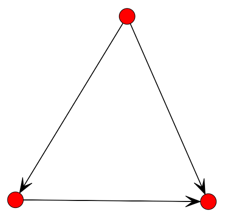
Here we show three usages of **PAFit**: the estimation of the attachment function A_k in isolation in Section 5.1, the estimation of node fitnesses η_i in isolation in Section 5.2, and the joint estimation of A_k and the η_i values in Section 5.3.

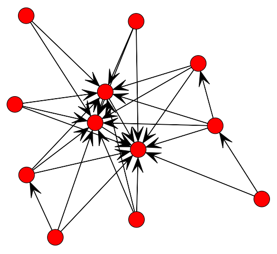
5.1. Attachment function estimation

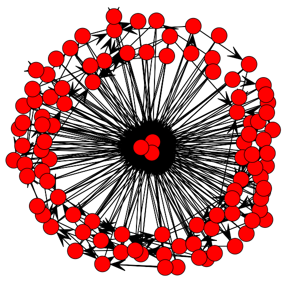
First we generate a network from a directed version of the BA model, called Price's model (de Solla Price 1976). From the initial graph with two nodes and one edge, one new node with $m = 5$ new edges is added at each time-step until the number of nodes is $N = 1000$.

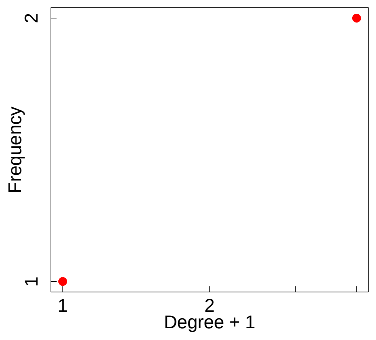
```
R> set.seed(1)
R> library("PAFit")
R> sim_net_1 <- generate_BA(N = 1000, m = 5)
```

Recall that A_k is linear in the BA model, i.e., the attachment exponent α is equal to 1, and the node fitnesses are uniform.

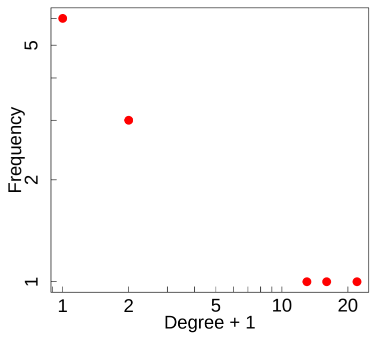

(a) Snapshot at $t = 1$.


(b) Snapshot at $t = 10$.

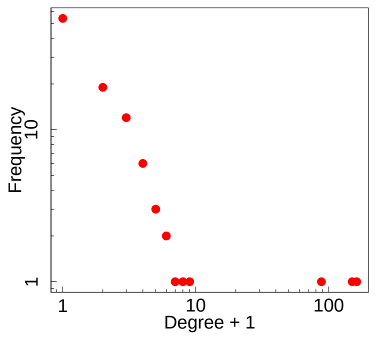

(c) Snapshot at $t = 100$.



(d) Degree distribution at $t = 1$.



(e) Degree distribution at $t = 10$.



(f) Degree distribution at $t = 100$.

Figure 1: Network snapshots and their corresponding in-degree distributions at time-steps $t = 1$, 10, and 100. The temporal network, `sim_net_1`, is generated from Price's model with total number of nodes $N = 1000$.

One can observe the emergence of hubs in this network by visualizing the generated graph at various time-steps by the function `plot`. The following code plots the network snapshot at time $t = 1$ in Figure 1a and its corresponding degree distribution in Figure 1d:

```
R> plot(sim_net_1, slice = 1, arrowhead.cex = 3, vertex.cex = 3)
R> plot(sim_net_1, slice = 1, plot = "degree", cex = 3, cex.axis = 2,
+    cex.lab = 2)
```

Note that if the network is directed, as it is in this example, the option `plot = "degree"` will plot the in-degree distribution. In the same way, we plot network snapshots at time $t = 10$ and $t = 100$ in Figures 1b and 1c and their corresponding degree distributions in Figures 1e and 1f.

The next step is to use the function `get_statistics` to get the summary statistics for the temporal network:

```
R> stats_1 <- get_statistics(sim_net_1)
```

With `stats_1` containing all the needed summary statistics, we then apply the three methods of estimating the attachment function in isolation:

```
R> result_Jeong <- Jeong(sim_net_1, stats_1)
R> result_Newman <- Newman(sim_net_1, stats_1)
R> result_PA_only <- only_A_estimate(sim_net_1, stats_1)
```

Let us explain `result_PA_only` in more detail. Information on the estimated results as well as the estimation process can be viewed by invoking `summary`:

```
R> summary(result_PA_only)

Estimation results by the PAFit method.
Mode: Only the attachment function was estimated.
Selected r parameter: 0.1
Estimated attachment exponent: 0.9820768
Attachment exponent ± 2 s.d.: (0.9720472,0.9921065)
------------------------------------------
Additional information:
Number of bins: 50
Number of iterations: 47
Stopping condition: 1e-08
```

As stated in Section 2, the PAFit method first finds the r parameter, which regularizes the PA function, by cross-validation, and then estimates A_k using the chosen r. The estimated function can be accessed via `$estimate_result$k` and `$estimate_result$A` of `result_PA_only`. From this estimated function, the attachment exponent α (when we assume $A_k = k^\alpha$) and its standard deviation are also estimated. Here $\hat{\alpha}$ is 0.98 ± 0.01 as we can see from the output of `summary`. These values can be accessed via `$estimate_result$alpha` and `$estimate_result$ci`.

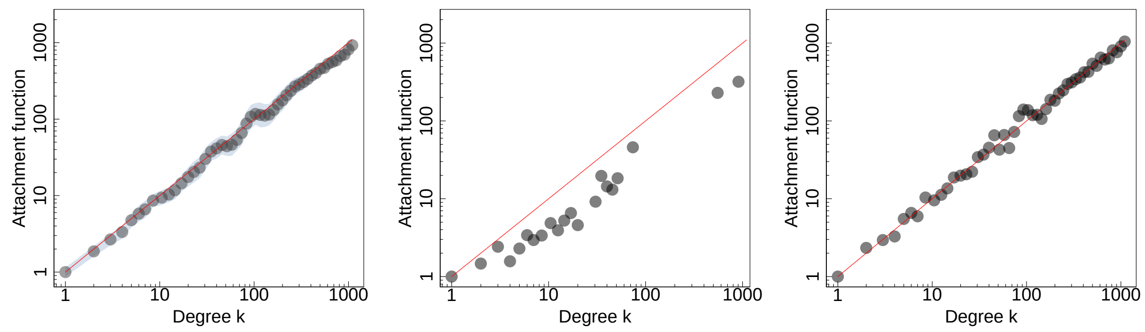
The output also reveals that **PAFit** applies binning with 50 bins by default. In this procedure, we divide the range of k into bins consisting of consecutive degrees, and assume that all k in a bin have the same value of A_k. Binning is an important regularization technique that significantly stabilizes the estimation of the attachment function (Pham *et al.* 2015). In this example, the center of each bin is stored in the field `$center_k` of `stats_1`.

Since the center of a bin is also the PA value corresponding to that bin in the linear PA case, we can plot the estimated attachment function together with the true attachment function using the following code, which produces the plot of Figure 2a. The options `min_A` and `max_A` specify the minimum and maximum values in the vertical axis of the plot, respectively.

```
R> plot(result_PA_only, stats_1, min_A = 1, max_A = 2000,
+    cex = 3, cex.axis = 2, cex.lab = 2)
R> lines(stats_1$center_k, stats_1$center_k, col = "red")
```

The estimation results of Jeong's method and Newman's method can be plotted in a similar way, and are shown in Figures 2b and 2c, respectively.

Overall, Newman's method and PAFit estimate the attachment function A_k about equally well, while Jeong's method is found to underestimate the function and also exhibits high variance. This can also be observed in the estimated attachment exponent of the three methods: Newman's method and PAFit estimate α reasonably well, while Jeong's method



(a) PAFit ($\hat{\alpha} = 0.98 \pm 0.01$)

(b) Jeong's method ($\hat{\alpha} = 0.89 \pm 0.09$)

(c) Newman's method ($\hat{\alpha} = 0.99 \pm 0.03$)

Figure 2: Estimating the attachment function in isolation from `sim_net_1`. The true attachment function is $A_k = k^\alpha$ with attachment exponent $\alpha = 1$. We also show the estimated α and the interval of the estimated $\alpha \pm 2$ s.d. provided by each method.

underestimates it. Note that in PAFit we also obtain the interval of the estimated $A_k \pm 2$ s.d. (lightblue region in Figure 2a), which is unavailable in the other two methods. This is a significant advantage of PAFit over the other two methods since it allows the user to quantify uncertainties in the result.

5.2. Node fitnesses estimation

Here we estimate node fitnesses from a BB model generated network with the assumption that $A_k = k$. To demonstrate the functionality of the package, we generate a BB network with a nonstandard setting:

```
R> set.seed(1)
R> sim_net_2 <- generate_BB(N = 1000, num_seed = 100, multiple_node = 100,
+    m = 15, s = 10)
```

This network grows from a seed network with $N_0 = 100$ nodes where the nodes form a line graph. The value of N_0 can be specified by `num_seed`. At each time-step we add $n = 100$ new nodes where each node has $m = 15$ new edges. The values of n and m can be specified via `multiple_node` and `m`, respectively. The total number of nodes in the final network is $N = 1000$. Finally, the distribution from which we generate node fitnesses is the Gamma distribution with mean 1 and inverse variance $s = 10$.

Next we get the network summary statistics and then apply the estimation function:

```
R> stats_2 <- get_statistics(sim_net_2)
R> result_fit_only <- only_F_estimate(sim_net_2, stats_2)
R> plot(result_fit_only, stats_2, plot = "f",
+    cex = 2, cex.axis = 1.5, cex.lab = 1.5)
```

The final line of the code snippet generates the distribution of estimated node fitnesses shown in Figure 3a.

The function `only_F_estimate` estimates node fitnesses under the assumption that $A_k = k$ by default. But one also can estimate node fitnesses in the Caldarelli model, i.e., $A_k = 1$ for all k, with the option `model_A = "Constant"`. The function `only_F_estimate` works by first finding the estimated value $\hat{s}$ of s by cross-validation, and then using $\hat{s}$ in the subsequent estimation of node fitnesses. The summary information of the estimation result can be viewed by invoking `summary`:

```
R> summary(result_fit_only)

Estimation results by the PAFit method.
Mode: Only node fitnesses were estimated.
Selected s parameter: 25
--------------------------------------------
Additional information:
Number of bins: 50
Number of iterations: 9
Stopping condition: 1e-08
```

The method slightly over-estimated s. We can check whether the node fitnesses were estimated well by plotting the estimated fitnesses versus the true fitnesses by running the following command:

```
R> plot(result_fit_only, stats_2, true_f = sim_net_2$fitness,
+    plot = "true_f", cex = 2, cex.axis = 1.5, cex.lab = 1.5)
```

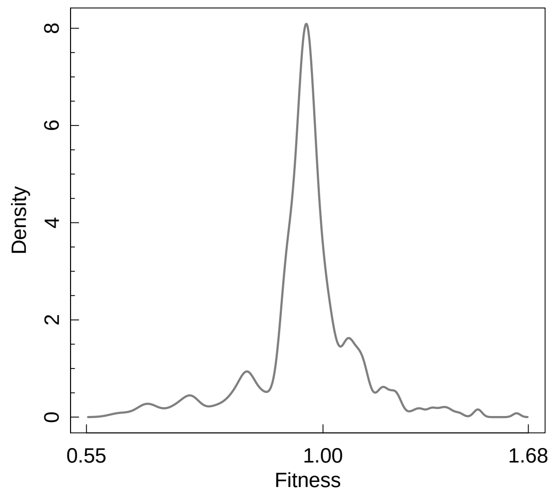
This will produce the plot of Figure 3b. It turns out that the estimated node fitnesses agree pretty well with the true node fitnesses. We note that the light blue band around the $\hat{\eta}_i$ values depicts the intervals of $\hat{\eta}_i \pm 2$ s.d. The upper and lower values can be accessed via `$estimate_result$upper_f` and `$estimate_result$lower_f` of `result_fit_only`, respectively.

5.3. Joint estimation of the attachment function and node fitnesses

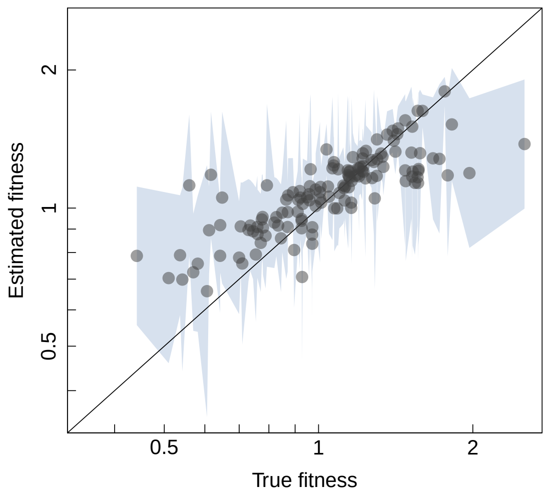
Here we show how to estimate the attachment function and node fitnesses simultaneously. We need to assume in Section 5.1 the equality of all η_i for the estimation of A_k in isolation, and in Section 5.2 a specific functional form of A_k for the estimation of η_i in isolation. Such assumptions become unnecessary when we perform joint estimation, since the appropriate functional forms will be automatically enforced through the regularization parameters r and s, which will be chosen by cross-validation. We recommend the joint estimation procedure as the standard estimation procedure in this package, unless there is a specific reason to justify the one or the other of these assumptions.

This time we generate a network in which the attachment function is $A_k = k^\alpha$ with $\alpha = 0.5$ and the Gamma distribution of node fitnesses has mean 1 and variance $1/s$ with $s = 10$:

```
R> set.seed(1)
R> sim_net_3 <- generate_net(N = 1000, num_seed = 100, multiple_node = 100,
+    m = 15, s = 10, alpha = 0.5)
```



(a) Distribution of estimated fitnesses.



(b) Estimated fitnesses versus true fitnesses.

Figure 3: Estimating node fitnesses in isolation from `sim_net_2`, which is generated with attachment function $A_k = k$. The true node fitnesses are sampled from a Gamma distribution with mean 1 and inverse variance 10. The attachment function in the estimation method is fixed at $A_k = k$. In panel b, we only plot nodes for which the number of acquired new edges is at least 5.

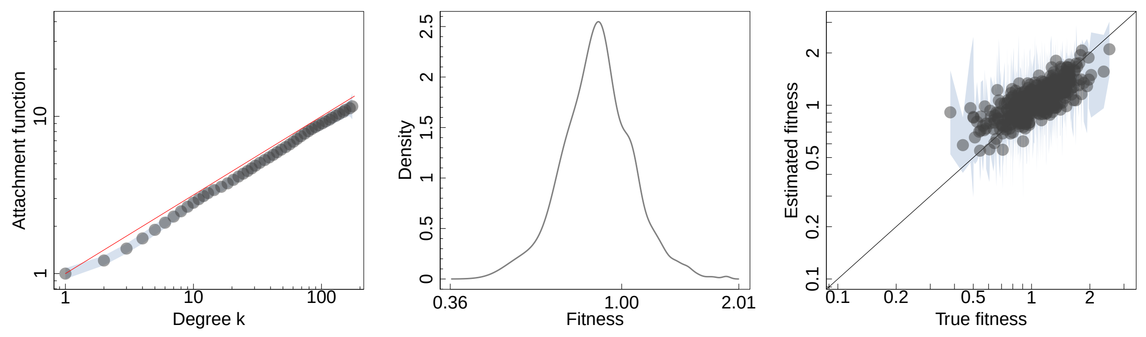
We then apply `joint_estimate`:

```
R> stats_3 <- get_statistics(sim_net_3)
R> result_PAFit <- joint_estimate(sim_net_3, stats_3)
R> summary(result_PAFit)
```

```
Estimation results by the PAFit method.
Mode: Both the attachment function and node fitness were estimated.
Selected r parameter: 8.75
Selected s parameter: 10
Estimated attachment exponent: 0.5030849
Attachment exponent ± 2 s.d.: (0.4966602,0.5095096)
------------------------------------------
Additional information:
Number of bins: 50
Number of iterations: 427
Stopping condition: 1e-08
```

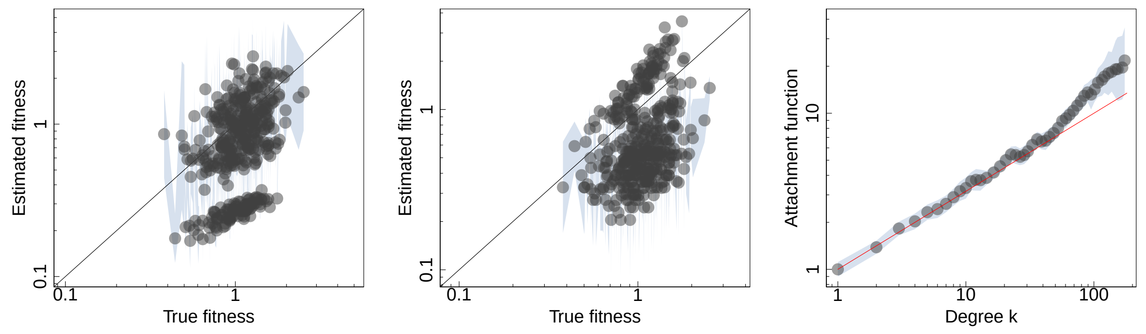
We can plot the estimated attachment function as in Figure 4a, and the distribution of the $\hat{\eta}_i$'s as in Figure 4b with the following code:

```
R> plot(result_PAFit, stats_3, min_A = 1, max_A = 40,
+    cex = 3, cex.axis = 2, cex.lab = 2)
R> lines(stats_3$center_k, stats_3$center_k^0.5, col = "red")
R> plot(result_PAFit, stats_3, plot = "f",
+    cex = 3, cex.axis = 2, cex.lab = 2)
```



(a) Estimated attachment function ($\hat{\alpha} = 0.50 \pm 0.01$).
(b) Distribution of estimated node fitnesses.
(c) Estimated and true η_i values.

Figure 4: Joint estimation of the attachment function and node fitnesses from `sim_net_3`. The red line in panel a is the true attachment function $A_k = k^{0.5}$. The true node fitnesses are sampled from a Gamma distribution with mean 1 and inverse variance $s = 10$.



(a) Estimated and true η_i values when assuming $A_k = k$. $\hat{s} = 2.7$.
(b) Estimated and true η_i values when assuming $A_k = 1$. $\hat{s} = 4$.
(c) Estimated PA function in isolation ($\hat{\alpha} = 0.57 \pm 0.02$).

Figure 5: Estimation of node fitnesses and the PA function in isolation from `sim_net_3`. The red line in panel c is the true attachment function $A_k = k^{0.5}$. The true node fitnesses are sampled from a Gamma distribution with mean 1 and inverse variance $s = 10$.

Concerning the estimated values, $\hat{s} = 10$ and $\hat{\alpha} = 0.50 \pm 0.01$ are good estimates of s and α, respectively. We can also plot the estimated fitnesses versus the true fitnesses as in Figure 4c with the following code:

```
R> plot(result_PAFit, stats_3, true_f = sim_net_3$fitness, plot = "true_f",
+    cex = 3, cex.axis = 2, cex.lab = 2)
```

We show how joint estimation improves on estimating either node fitnesses in isolation (Figures 5a and 5b) or the PA function in isolation (Figure 5c). For estimating node fitnesses in isolation, two cases are shown: the result when we assume the BB model in which $A_k = k$ (Figure 5a) and the result when we assume the Caldarelli model in which $A_k = 1$ (Figure 5b). In either case, the estimated node fitnesses are visually worse than those of the joint estimation in Figure 4c. Similarly, estimating the PA function in isolation apparently led to overestimation of the PA function in the region of large k. To conclude, estimating either node fitnesses or the PA function in isolation would likely be worse than the joint estimation, if the underlying assumptions about the true node fitnesses and the true PA function are wrong.

6. Simulation study

In this section, we present the results of a simulation study that we conducted to assess the performance of the `joint_estimate` function. We assume the functional form $A_k = k^\alpha$ for the attachment function. To cover the spectrum of PA and anti-PA phenomena, we choose four values for α: -0.5, 0, 0.5, and 1. We sample node fitnesses from a Gamma distribution with mean 1 and variance $1/s$. Three values for s were chosen: 5, 20, and 80. While small values of s lead to widely varied node fitnesses, large values of s leads to highly concentrated node fitnesses.

For each combination of α and s, we generated $M = 50$ networks, and estimated A_k and s for each network using the `joint_estimate` function. We then fit the form $\hat{A}_k = k^\alpha$ to $\hat{A}_k$ in

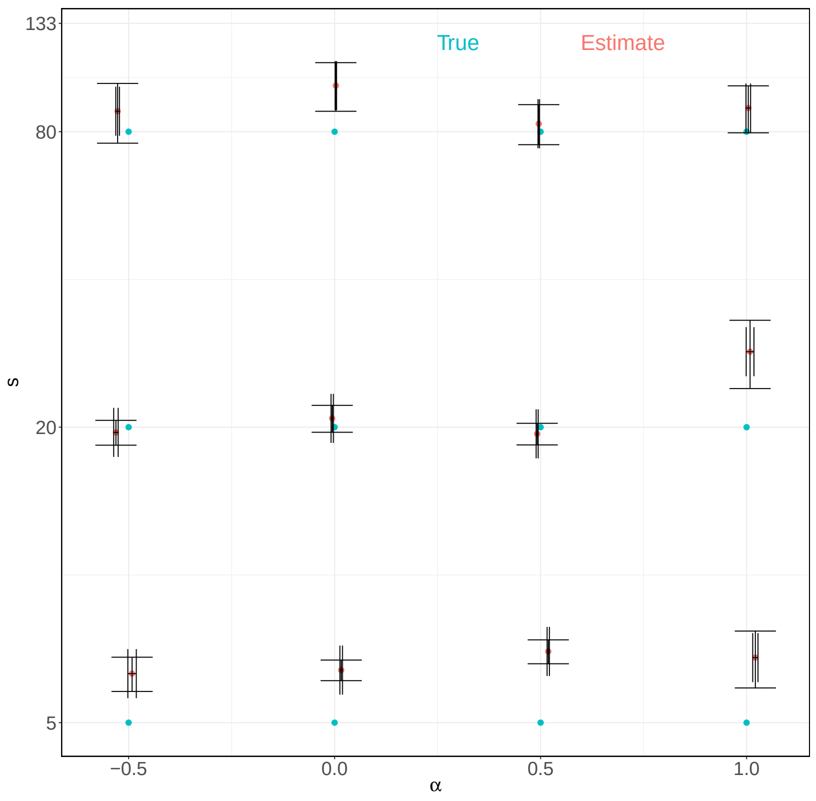


Figure 6: Simulation result. For each combination of α and s, we generated 50 networks using attachment function $A_k = k^\alpha$ and a Gamma distribution with mean 1 and inverse variance s for node fitnesses. Each red point indicates the average of the esimated α and the selected s over 100 simulations for the corresponding combination of α and s. At each red point, the horizontal/vertical bar indicates the interval of the estimated α/selected $s \pm 2$ s.d., respectively.

order to estimate α. We then compared the means of the estimation results of α and s with the true values. Each simulated network has a total of 1000 nodes, where the initial graph has 200 nodes and 50 new nodes are added at each time-step for a total of 10 time-steps. Each new node has 50 new edges.

The results are shown in Figure 6. The attachment exponent α was estimated reasonably well across all combinations of α and s. Except for the cases in which the attachment function grows fast ($\alpha = 1$) or the case in which node fitnesses have high variance ($s = 5$), the estimated values of s were also acceptable. The case of $s = 5$ resulted in a slight over-estimation, which is perhaps attributable to high node fitnesses variance. We also notice that s was slightly over-estimated when $\alpha = 1$, which may be caused by the fast growing rate of the PA function. One also notices that the intervals of $\hat{s} \pm 2$ s.d. are much larger than those for $\hat{\alpha}$. The above observations imply that it is much harder to estimate s than α.

7. Analysis of a collaboration network between scientists

In this section, we demonstrate the complete work-flow for the joint estimation of A_k and η_i on a collaboration network between scientists from the field of complex networks. In this network, nodes represent scientists and an undirected edge exists between them if, and only if, they have coauthored a paper. The degree of a node represents the number of collaborators of a scientist, since multiple edges are not considered. The temporal network is stored in `coauthor.net`, and the names of the scientists are stored in `coauthor.author_id`. The network without timestamps was compiled by Mark Newman from the bibliographies of two review articles on complex networks (Newman 2006). Paul Sheridan, the second author of the present work, augmented the dataset with timestamps. More information on the dataset can be found in the package reference manual.

The first step in the analysis is to convert the edge-list matrix `coauthor.net` to a '`PAFit_net`' object, and get the summary statistics using the function `get_statistics`.

```
R> set.seed(1)
R> true_net <- as.PAFit_net(coauthor.net, type = "undirected")
R> net_stats <- get_statistics(true_net)
R> summary(net_stats)
```

```
Contains summary statistics for the temporal network.
Type of network: undirected
Number of nodes in the final network: 1498
Number of edges in the final network: 2849
Number of new nodes: 1358
Number of new edges: 1255
Number of time-steps: 145
Maximum degree: 37
Number of bins: 38
```

The temporal network grew in 145 time-steps from an initial network at September 2000, to a final state at September 2007. The resolution of those time-steps is monthly. The final network has 1498 scientists with 2849 collaborations among them. One can plot the degree distribution of the final snapshot as follows:

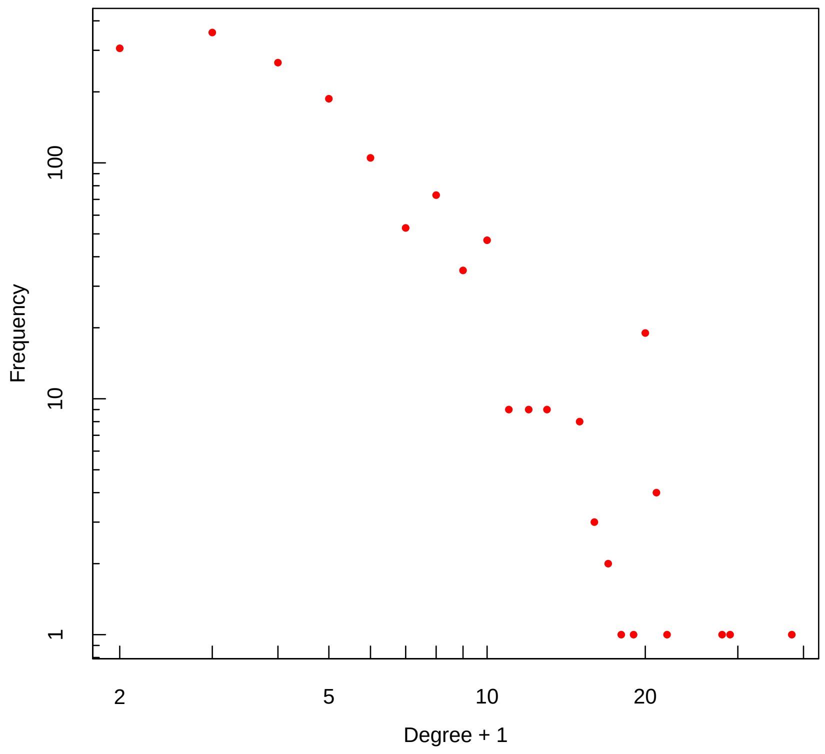


Figure 7: Degree distribution of the final snapshot of the collaboration network.

```
R> plot(true_net, plot = "degree")
```

This will produce the plot of Figure 7.

Before any estimation of the PA function and node fitnesses is carried out, one can test whether the linear PA-only case is consistent with the observed degree distribution of the collaboration network:

```
R> test_linear_PA_result <- test_linear_PA(net_stats$final_deg)
R> print(test_linear_PA_result)
```

This will generate Table 5. In this case, since the Yule and Waring distributions are not the best models, one can conclude that the linear PA-only case is inconsistent with the observed degree vector.

To further investigate the PA function and node fitnesses, we invoke the `joint_estimate` function for joint estimation:

```
R> full_result <- joint_estimate(true_net, net_stats)
R> summary(full_result)
```

```
Estimation results by the PAFit method.
Mode: Both the attachment function and node fitness were estimated.
Selected r parameter: 10
Selected s parameter: 45
```

Model	Log-likelihood	AIC	BIC
nb	−2415.08	4840.16	4866.72
pois	−2468.63	4945.27	4966.51
waring	−2557.49	5124.99	5151.55
geom	−2898.64	5811.27	5848.45
yule	−2959.97	5929.95	5956.51

Table 5: The result of applying the function `test_linear_PA` to the observed degree vector of the collaboration network. This function calculates the AIC and BIC of five models: Yule (`yule`), Waring (`waring`), Poisson (`pois`), geometric (`geom`), and negative binomial (`nb`) when fitting them to the observed degree vector.

```
Estimated attachment exponent: 0.9951764
Attachment exponent ± 2 s.d.: (0.9715202,1.018833)
------------------------------------------
Additional information:
Number of bins: 38
Number of iterations: 607
Stopping condition: 1e-08
```

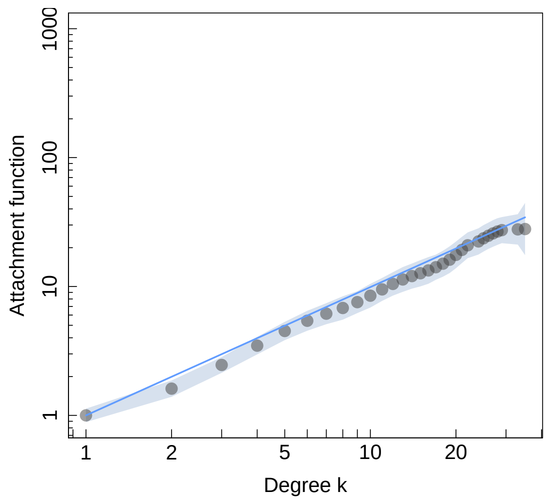
We can visualize the estimated attachment function and the distribution of estimated node fitnesses by:

```
R> plot(full_result, net_stats, line = TRUE,
+    cex = 2, cex.axis = 1.5, min_A = 1, max_A = 1000, cex.lab = 1.5)
R> plot(full_result, net_stats, plot = "f",
+    cex = 2, cex.axis = 1.5, cex.lab = 1.5)
```

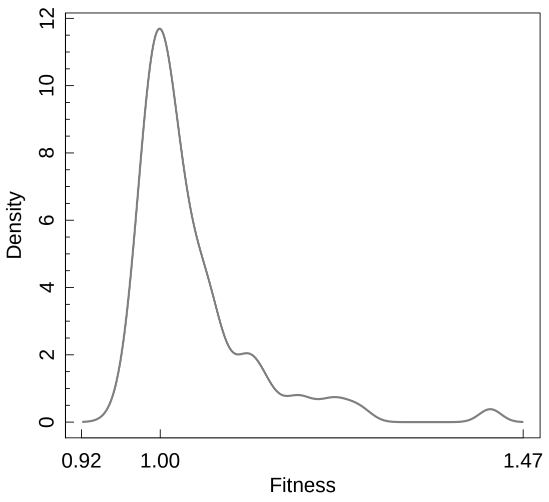
This code will sequentially generate the plots of Figures 8a and 8b. When other options are set at their default values, the option `line = TRUE` will plot the function $\hat{A}_k = k^{\hat{\alpha}}$, which is a straight line on a double logarithmic scale.

The best fit model when we performed joint estimation is close to the BB model. In Figure 8a, the estimated A_k is an increasing function with $\hat{\alpha} = 1.00 \pm 0.02$. We take this as evidence in favor of the presence of linear PA in the collaboration network. Let us take a concrete example: A network scientist with twenty collaborators has roughly twice the chance to get a new collaborator compared with someone who only has ten collaborators, assuming they have the same fitness. For comparison's sake, we also plot the estimation results of A_k in isolation using Jeong's method, Newman's method, and PAFit in Figures 8c, 8d, and 8e, respectively:

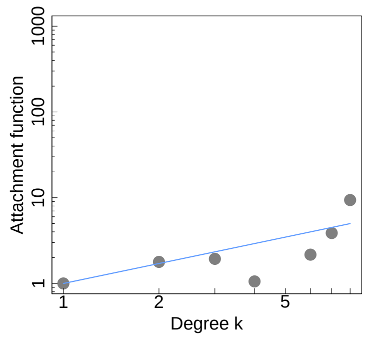
```
R> result_Jeong <- Jeong(true_net, net_stats)
R> result_Newman <- Newman(true_net, net_stats)
R> result_onlyA <- only_A_estimate(true_net, net_stats)
R> plot(result_Jeong, net_stats, line = TRUE, min_A = 1, max_A = 1000,
+    cex = 3, cex.axis = 2, cex.lab = 2)
R> plot(result_Newman, net_stats, line = TRUE, min_A = 1, max_A = 1000,
+    cex = 3, cex.axis = 2, cex.lab = 2)
R> plot(result_onlyA, net_stats, line = TRUE, min_A = 1, max_A = 1000,
+    cex = 3, cex.axis = 2, cex.lab = 2)
```



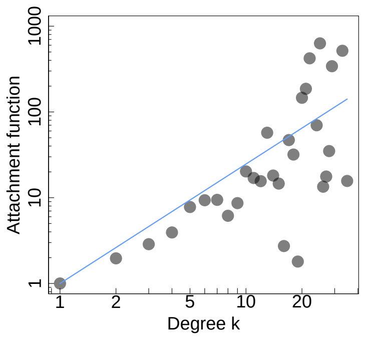
(a) Estimated A_k with joint estimation by PAFit ($\hat{\alpha} = 1.00 \pm 0.02$).



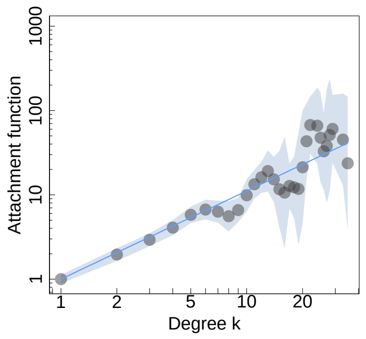
(b) Histogram of estimated node fitnesses.



(c) Jeong's method ($\hat{\alpha} = 0.77 \pm 0.80$).



(d) Newman's method ($\hat{\alpha} = 1.39 \pm 0.56$).



(e) PAFit ($\hat{\alpha} = 1.05 \pm 0.07$).

Figure 8: Estimation of the attachment function and node fitnesses for the network scientist collaboration network. Panels a and b show the joint estimation result, while panels c, d, and e show the results when we estimated the PA function in isolation.

The options `min_A = 1` and `max_A = 1000` specify the range of the vertical axis and are needed for making the plots comparable.

The high variance of $\hat{\alpha}$ from either Jeong's method or Newman's method would render qualitative assessments of the PA function inconclusive, if one relied only on those methods: One could not confidently ascertain whether the PA function is sub-linear, linear, or super-linear in nature. We notice that the estimated A_k obtained from the joint estimation resembles that of Figure 8e, when we estimate it in isolation. The reason is that estimated node fitnesses in Figure 8b are highly concentrated around the mean. Thus their distribution is not very far from the case when all the fitnesses are 1. Nevertheless, we observe that the estimated A_k from the joint estimation is reduced when compared with that of Figure 8e. This is expected since in the joint estimation, a portion of the connection probability in Equation 1 is explained by node fitness.

Rank	Estimated fitness	Name
1	1.42	BARABASI, A
2	1.35	NEWMAN, M
3	1.26	JEONG, H
4	1.25	LATORA, V
5	1.24	ALON, U
6	1.23	OLTVAI, Z
7	1.23	YOUNG, M
8	1.22	WANG, B
9	1.21	SOLE, R
10	1.21	BOCCALETTI, S

Table 6: Ten topmost "fittest" network scientists in the field of complex networks.

Although the distribution in the plot of Figure 8b is concentrated around its mean, we notice that its right tail is rather long, which is a sign that this tail contains interesting information. We can extract the information from this region by finding the topmost "fittest" network scientists. This can be done as follows:

```
R> sorted_fit <- sort(full_result$estimate_result$f, decreasing = TRUE)
R> top_scientist <- coauthor.author_id[names(sorted_fit), ]
R> print(cbind(sorted_fit[1:10], top_scientist[1:10, 2]))
```

This code snippet will produce the results show in Table 6. The table shows the ten network scientists that we found to have the highest ability to attract new collaborators from the field. Anyone acquainted with the field will recognize a number of familiar names. For example, at the top of the list is none other than Albert-László Barabási, who introduced the BA model. Number two and number three are Mark Newman and Hawoong Jeong, who respectively are the authors of the eponymously named Newman's method and Jeong's method.

8. Conclusion

We introduced the R package **PAFit**, which provides a comprehensive framework for the non-parametric estimation of PA and node fitness mechanisms in the growth of temporal complex networks. In summary, **PAFit** implements functions to simulate various temporal network models based on these two mechanisms, gathers summary statistics from real-world temporal network datasets, and estimates non-parametrically the attachment function and node fitnesses. We provided a number of simulated examples, as well as a complete analysis of a real-world collaboration network.

Acknowledgments

This work was supported in part by grants from the Japan Society for the Promotion of Science KAKENHI [JP16J03918, JP19K20231 to T.P. and JP16H01547 to H.S.].

References

Akaike H (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control*, **19**(6), 716–723. doi:10.1109/tac.1974.1100705.

Albert R, Barabási AL (1999). "Emergence of Scaling in Random Networks." *Science*, **286**(5439), 509–512. doi:10.1126/science.286.5439.509.

Albert R, Jeong H, Barabási A (2000). "Error and Attack Tolerance of Complex Networks." *Nature*, **406**(6794), 378–382. doi:10.1038/35019019.

Allison PD (1980). "Estimation and Testing for a Markov Model of Reinforcement." *Sociological Methods & Research*, **8**(4), 434–453. doi:10.1177/004912418000800405.

Barabási AL, Albert R, Jeong H (2000). "Scale-Free Characteristics of Random Networks: The Topology of The World-Wide Web." *Physica A: Statistical Mechanics and Its Applications*, **281**(1–4), 69–77. doi:10.1016/s0378-4371(00)00018-2.

Bender-deMoll S, Morris M (2019). ***tsna**: Tools for Temporal Social Network Analysis.* R package version 0.3.0, URL https://CRAN.R-project.org/package=tsna.

Bezáková I, Kalai A, Santhanam R (2006). "Graph Model Selection Using Maximum Likelihood." In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 105–112. ACM, New York, NY, USA. doi:10.1145/1143844.1143858.

Bianconi G, Barabási AL (2001). "Competition and Multiscaling in Evolving Networks." *Europhysics Letters*, **54**(4), 436–442. doi:10.1209/epl/i2001-00260-6.

Borgs C, Chayes J, Daskalakis C, Roch S (2007). "First to Market is Not Everything: An Analysis of Preferential Attachment with Fitness." In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, pp. 135–144. doi:10.1145/1250790.1250812.

Box-Steffensmeier JM, De Boef S (2006). "Repeated Events Survival Models: The Conditional Frailty Model." *Statistics in Medicine*, **25**(20), 3518–3533. doi:10.1002/sim.2434.

Butts CT (2019). ***sna**: Tools for Social Network Analysis.* R package version 2.5, URL https://CRAN.R-project.org/package=sna.

Butts CT, Leslie-Cook A, Krivitsky PN, Bender-deMoll S (2019). ***networkDynamic**: Dynamic Extensions for Network Objects.* R package version 0.10.0, URL https://CRAN.R-project.org/package=networkDynamic.

Caldarelli G (2007). *Scale-Free Networks.* Oxford Universiy Press.

Caldarelli G, Capocci A, De Los Rios P, Muñoz MA (2002). "Scale-Free Networks from Varying Vertex Intrinsic Fitness." *Physical Review Letters*, **89**(25), 258702. doi:10.1103/physrevlett.89.258702.

Callaway DS, Hopcroft JE, Kleinberg JM, Newman MEJ, Strogatz SH (2001). "Are Randomly Grown Graphs Really Random?" *Physical Review E*, **64**(4), 041902. doi:10.1103/physreve.64.041902.

Clauset A, Shalizi CR, Newman MEJ (2009). "Power-Law Distributions in Empirical Data." *SIAM Review*, **51**(4), 661–703. doi:10.1137/070710111.

Coleman JS (1964). *Introduction to Mathematical Sociology.* Free Press of Glencoe London.

Csardi G, Nepusz T (2006). "The **igraph** Software Package for Complex Network Research." *InterJournal*, **Complex Systems**, 1695. URL http://igraph.org/.

de Solla Price DJ (1976). "A General Theory of Bibliometric and Other Cumulative Advantage Processes." *Journal of the American Society for Information Science*, **27**(5), 292–306. doi:10.1002/asi.4630270505.

Dorogovtsev SN, Mendes JFF (2003). *Evolution of Networks: From Biological Nets to the Internet and WWW.* Oxford University Press, New York.

Dunne JA, Williams RJ, Martinez ND (2002). "Food-Web Structure and Network Theory: The Role of Connectance and Size." *Proceedings of the National Academy of Sciences of the United States of America*, **99**(20), 12917–12922. doi:10.1073/pnas.192407699.

Eddelbuettel D (2013). *Seamless R and C++ Integration with **Rcpp**.* Springer-Verlag, New York.

Eddelbuettel D, Balamuta JJ (2017). "Extending R with C++: A Brief Introduction to **Rcpp**." *PeerJ Preprints*, **5**, e3188v1. doi:10.7287/peerj.preprints.3188v1.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Eom YH, Jo HH (2014). "Generalized Friendship Paradox in Complex Networks: The Case of Scientific Collaboration." *Scientific Reports*, **4**, 4603. doi:10.1038/srep04603.

Erdös P, Rényi A (1959). "On Random Graphs." *Publicationes Mathematicae Debrecen*, **6**, 290–297.

Feld SL (1991). "Why Your Friends Have More Friends Than You Do." *American Journal of Sociology*, **96**(6), 1464–1477. doi:10.1086/229693.

Guetz AN, Holmes SP (2011). "Adaptive Importance Sampling for Network Growth Models." *The Annals of Operations Research*, **189**(1), 187–203. doi:10.1007/s10479-010-0685-2.

Handcock MS, Hunter DR, Butts CT, Goodreau SM, Krivitsky PN, Bender-deMoll S, Morris M (2019). ***statnet**: Software Tools for the Statistical Analysis of Network Data.* The Statnet Project (http://www.statnet.org/). R package version 2019.6, URL https://CRAN.R-project.org/package=statnet.

Handcock MS, Hunter DR, Butts CT, Goodreau SM, Krivitsky PN, Morris M (2018). ***ergm**: Fit, Simulate and Diagnose Exponential-Family Models for Networks.* The Statnet Project (http://www.statnet.org/). R package version 3.9.4, URL https://CRAN.R-project.org/package=ergm.

Handcock MS, Hunter DR, Butts CT, Goodreau SM, Morris M (2008). "**statnet**: Software Tools for the Representation, Visualization, Analysis and Simulation of Network Data." *Journal of Statistical Software*, **24**(1), 1–11. doi:10.18637/jss.v024.i01.

Handcock MS, Jones JH (2004). "Likelihood-Based Inference for Stochastic Models of Sexual Network Formation." *Theoretical Population Biology*, **65**(4), 413–422. doi:10.1016/j.tpb.2003.09.006.

Hunter DR, Handcock MS, Butts CT, Goodreau SM, Morris M (2008). "**ergm**: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks." *Journal of Statistical Software*, **24**(3), 1–29. doi:10.18637/jss.v024.i03.

Hunter DR, Lange K (2000). "Quantile Regression via an MM Algorithm." *Journal of Computational and Graphical Statistics*, **9**(1), 60–77. doi:10.2307/1390613.

Hunter DR, Lange K (2004). "A Tutorial on MM Algorithms." *The American Statistician*, **58**(1), 30–37. doi:10.1198/0003130042836.

INRA, Leger JB (2015). ***blockmodels**: Latent and Stochastic Block Model Estimation by a V-EM Algorithm.* R package version 1.1.1, URL https://CRAN.R-project.org/package=blockmodels.

Irwin JO (1963). "The Place of Mathematics in Medical and Biological Statistics." *Journal of the Royal Statistical Society A*, **126**(1), 1–45. doi:10.2307/2982445.

Jeong H, Néda Z, Barabási AL (2003). "Measuring Preferential Attachment in Evolving Networks." *Europhysics Letters*, **61**(4), 567–572. doi:10.1209/epl/i2003-00166-9.

Kelly PJ, Lim LLY (2000). "Survival Analysis for Recurrent Event Data: An Application to Childhood Infectious Diseases." *Statistics in Medicine*, **19**(1), 13–33. doi:10.1002/(sici)1097-0258(20000115)19:1<13::aid-sim279>3.0.co;2-5.

Kong JS, Sarshar N, Roychowdhury VP (2008). "Experience Versus Talent Shapes the Structure of the Web." *Proceedings of the National Academy of Sciences of the United States of America*, **105**(37), 13724–13729. doi:10.1073/pnas.0805921105.

Krapivsky PL, Rodgers GJ, Redner S (2001). "Organization of Growing Networks." *Physical Review E*, **63**, 066123. doi:10.1103/physreve.63.066123.

Krivitsky P, Handcock M (2008). "Fitting Latent Cluster Models for Networks with **latentnet**." *Journal of Statistical Software*, **24**(5), 1–23. doi:10.18637/jss.v024.i05.

Krivitsky PN, Handcock MS (2018). ***latentnet**: Latent Position and Cluster Models for Statistical Networks.* The Statnet Project (http://www.statnet.org/). R package version 2.9.0, URL https://CRAN.R-project.org/package=latentnet.

Krivitsky PN, Handcock MS (2019). ***tergm**: Fit, Simulate and Diagnose Models for Network Evolution Based on Exponential-Family Random Graph Models.* The Statnet Project (http://www.statnet.org). R package version 3.6.1, URL https://CRAN.R-project.org/package=tergm.

Kunegis J (2013). "KONECT – The Koblenz Network Collection." URL http://konect.uni-koblenz.de/.

Kunegis J, Blattner M, Moser C (2013). "Preferential Attachment in Online Networks: Measurement and Explanations." In *Proceedings of the 5th Annual ACM Web Science Conference*, WebSci '13, pp. 205–214. ACM, New York. doi:10.1145/2464464.2464514.

Leifeld P, Cranmer S, Desmarais B (2018). "Temporal Exponential Random Graph Models with **btergm**: Estimation and Bootstrap Confidence Intervals." *Journal of Statistical Software*, **83**(6), 1–36. doi:10.18637/jss.v083.i06.

Leskovec J, Krevl A (2014). "SNAP Datasets: Stanford Large Network Dataset Collection." http://snap.stanford.edu/data.

Liljeros F, Edling CR, Amaral LAN, Stanley HE, Aberg Y (2001). "The Web of Human Sexual Contacts." *Nature*, **411**(6840), 907–908. doi:10.1038/35082140.

Matias C, Miele V (2016). "Statistical Clustering of Temporal Networks through a Dynamic Stochastic Block Model." *Journal of the Royal Statistical Society B*, **79**(4), 1119–1141. doi:10.1111/rssb.12200.

Matias C, Miele V (2019). ***dynsbm***: *Dynamic Stochastic Block Models*. R package version 0.6, URL https://CRAN.R-project.org/package=dynsbm.

Momeni N, Rabbat MG (2015). "Measuring the Generalized Friendship Paradox in Networks with Quality-Dependent Connectivity." In *Complex Networks VI: Proceedings of the 6th Workshop on Complex Networks CompleNet 2015*, pp. 45–55. Springer-Verlag, Cham. doi:10.1007/978-3-319-16112-9_5.

Nekovee M, Moreno Y, Bianconi G, Marsili M (2007). "Theory of Rumour Spreading in Complex Social Networks." *Physica A: Statistical Mechanics and its Applications*, **374**(1), 457–470. doi:10.1016/j.physa.2006.07.017.

Newman MEJ (2001). "Clustering and Preferential Attachment in Growing Networks." *Physical Review E*, **64**(2), 025102(R). doi:10.1103/physreve.64.025102.

Newman MEJ (2006). "Finding Community Structure in Networks Using the Eigenvectors of Matrices." *Physical Review E*, **74**(3), 036104. doi:10.1103/physreve.74.036104.

Newman MEJ (2010). *Networks: An Introduction*. Oxford University Press, New York.

Newman MEJ, Forrest S, Balthrop J (2002). "Email Networks and the Spread of Computer Viruses." *Physical Review E*, **66**(3), 035101(R). doi:10.1103/physreve.66.035101.

Pastor-Satorras R, Vespignani A (2001). "Epidemic Spreading in Scale-Free Networks." *Physical Review Letters*, **86**(14), 3200–3203. doi:10.1103/physrevlett.86.3200.

Pham T, Sheridan P, Shimodaira H (2015). "PAFit: A Statistical Method for Measuring Preferential Attachment in Temporal Complex Networks." *PLOS ONE*, **10**(9), e0137796. doi:10.1371/journal.pone.0137796.

Pham T, Sheridan P, Shimodaira H (2016). "Joint Estimation of Preferential Attachment and Node Fitness in Growing Complex Networks." *Scientific Reports*, **6**, 32558. doi:10.1038/srep32558.

Pham T, Sheridan P, Shimodaira H (2020). ***PAFit***: *Generative Mechanism Estimation in Temporal Complex Networks*. R package version 1.0.1.8, URL https://CRAN.R-project.org/package=PAFit.

Raftery AE (1995). “Bayesian Model Selection in Social Research.” *Sociological Methodology*, **25**, 111–163. doi:10.2307/271063.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Redner S (2005). “Citation Statistics from 110 Years of Physical Review.” *Physics Today*, **58**(6), 49–54. doi:10.1063/1.1996475.

Ripley RM, Snijders TAB, Bóda Z, Vörös A, Preciado P (2018). “Manual for Siena Version 4.0.” *Technical report*, University of Oxford, Department of Statistics; Nuffield College, Oxford. R package version 1.2-12, URL https://CRAN.R-project.org/package=RSiena.

Ronda-Pupo GA, Pham T (2018). “The Evolutions of the Rich Get Richer and the Fit Get Richer Phenomena in Scholarly Networks: The Case of the Strategic Management Journal.” *Scientometrics*, **116**(1), 363–383. doi:10.1007/s11192-018-2761-3.

Schweinberger M, Luna P (2018). “**hergm**: Hierarchical Exponential-Family Random Graph Models.” *Journal of Statistical Software*, **85**(1), 1–39. doi:10.18637/jss.v085.i01.

Sheridan P, Kamimura T, Shimodaira H (2010). “A Scale-Free Structure Prior for Graphical Models with Applications in Functional Genomics.” *PLoS ONE*, **5**(11), e13580. doi:10.1371/journal.pone.0013580.

Sheridan P, Onodera T (2018). “A Preferential Attachment Paradox: How Preferential Attachment Combines with Growth to Produce Networks with Log-Normal In-Degree Distributions.” *Scientific Reports*, **8**(1), 2811. doi:10.1038/s41598-018-21133-2.

Simon HA (1955). “On a Class of Skew Distribution Functions.” *Biometrika*, **42**(3–4), 425–440. doi:10.1093/biomet/42.3-4.425.

Sinatra R, Wang D, Deville P, Song C, Barabási AL (2016). “Quantifying the Evolution of Individual Scientific Impact.” *Science*, **354**(6312). doi:10.1126/science.aaf5239.

Wang D, Song C, Barabási AL (2013). “Quantifying Long-Term Scientific Impact.” *Science*, **342**(6154), 127–132. doi:10.1126/science.1237825.

Yule GU (1925). “A Mathematical Theory of Evolution, Based on the Conclusions of Dr. J.C. Willis, F.R.S.” *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, **213**(402–410), 21–87. doi:10.1098/rstb.1925.0002.

Zhou H, Alexander D, Lange K (2011). “A Quasi-Newton Acceleration for High-Dimensional Optimization Algorithms.” *Statistics and Computing*, **21**(2), 261–273. doi:10.1007/s11222-009-9166-3.

Affiliation:

Thong Pham
Mathematical Statistics Team
RIKEN Center for Advanced Intelligence Project
Tokyo, Japan
E-mail: `thong.pham@riken.jp`

Journal of Statistical Software `http://www.jstatsoft.org/`
published by the Foundation for Open Access Statistics `http://www.foastat.org/`
February 2020, Volume 92, Issue 3 *Submitted:* 2017-04-19
`doi:10.18637/jss.v092.i03` *Accepted:* 2019-04-28