PAFit: an R Package for Estimating Preferential Attachment and Node Fitness in Temporal Complex Networks

Thong Pham
RIKEN Center for Advanced Intelligence Project

Paul Sheridan
Hirosaki University

Hidetoshi Shimodaira
Kyoto University
RIKEN Center for Advanced Intelligence Project

Abstract

Many real-world systems are profitably described as complex networks that grow over time. Preferential attachment and node fitness are two ubiquitous 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 standard statistical packages for estimating the structural properties of complex networks, there is no corresponding package when it comes to the estimation of growth mechanisms. This paper introduces the R package **PAFit**, which implements well-established statistical methods for estimating preferential attachment and node fitness, 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 good performance for large-scale networks. In this paper, we first introduce the main functionalities of **PAFit** using simulated examples, and then use the package to analyze a collaboration network between scientists in the field of complex networks.

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 *et al.* 2000), and sexual relation networks between members of a community (Liljeros *et al.* 2001). One driver of the popularity of this modeling paradigm is that complex networks let us abstract away domain-dependent details and focus on modeling important structural properties observed in real-world systems, in the hope that we will be able to predict or control the future behavior of such systems.

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 most dynamical network processes must travel frequently through highly-

connected nodes, this understanding in turns sheds light on the answers of important practical questions, including how to prevent the spreading of rumors (Nekovee *et al.* 2007), how to stop a virus outbreak (Pastor-Satorras and Vespignani 2001), and how to guard against cybernetic attacks (Albert *et al.* 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 $k^{-\gamma}$ for $2 < \gamma \leq 3$. 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 *et al.* 2002), stretched exponential (Newman *et al.* 2002), and power-law with exponential cut-off (Clauset *et al.* 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 prompted the network scientists to search for new modeling ingredients capable of explaining heavy-tailed degree distributions. It turns out that temporal complex network models that incorporate 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 a 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 (Bianconni and Barabási 2001). Growth mechanisms, which governs how a node acquires new edges in the growth process, are the most important elements that distinguishes different temporal network models.

This paper focuses on estimating two interpretable and ubiquitous growth mechanisms: preferential attachment (PA) and node fitness. While they are based on simple concepts that are shared in diverse fields, they are also flexible enough to produce a wide range of different networks. In the PA mechanism, the probability P_i a node v_i gets a new edge in the future is proportional to some positive function A_{k_i} of its current degree k_i. This function is called the attachment function.

The name 'preferential attachment' stems from the motivation for the mechanism: if A_k is an increasing function on average, 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, we will say that preferential attachment exists if A_k is an increasing function on average. We recover the BA model in the special case when $A_k = k$. This functional form in fact has been long known in other fields with various under names such as 'rich-get-richer' (Simon 1955) and 'cummulative advantages' (Price 1976). When A_k assumes the log-linear form of k^α, with $\alpha > 0$ called the attachment exponent, we have the generalized BA model (Krapivsky *et al.* 2001).

While P_i depends on the degree of v_i in the PA mechanism, in the fitness mechanism P_i depends only on a positive quantity η_i called the fitness of node v_i. We can interpret η_i as the intrinsic attractiveness of v_i. The fitness mechanism offers a simple way to express the variance in edge-acquiring abilities between nodes with 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 intrinsic fitnesses.

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

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

As we will show in Section 2, (1) encompasses many well-known temporal network models. Based on the functional form of A_k and the distribution of η_i, the model depicted in (1) can produce networks with various degree distributions (Bianconni and Barabási 2001; Caldarelli *et al.* 2002; Borgs *et al.* 2007; Kong *et al.* 2008).

There are implementations of standard statistical methodologies for estimating complex network degree distribution, for example the R packages **igraph** (Csardi and Nepusz 2006) and **poweRlaw** (Gillespie 2015), but the corresponding standard methods for estimating the underlying growth mechanisms are not implemented anywhere. This is unsatisfactory because PA (and other growth mechanisms) and degree distribution are a package deal in so far as temporal complex networks are concerned.

A growing number of interesting applications also call for an implementation of PA and node fitness estimation methods. Based on the functional forms of A_k and η_i, we can check whether two important social phenomena called 'rich-get-richer' or 'fit-get-richer' exist in a temporal network (Pham *et al.* 2016). The two mechanisms have also been proposed to be the underlying mechanisms of another important 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 *et al.* 2010; Guetz and Holmes 2011), World Wide Web (Kong *et al.* 2008), Internet topology graphs (Bezáková *et al.* 2006), and citation networks (Wang *et al.* 2013; Sinatra *et al.* 2016). Finally, we can classify real-world temporal network data based on the estimated attachment exponent of A_k (Kunegis *et al.* 2013).

This paper introduces the R package **PAFit** (Pham *et al.* 2017), which fills the gap with an implementation of the standard PA and node fitness estimation procedures. In particular, we implement Jeong's method (Jeong *et al.* 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 in estimating the attachment function A_k in isolation, while the last one is a principled statistical method that can either estimate 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.

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 will discuss how **PAFit** relates with exisiting network analysis packages in Section 4.

In Section 6, the package is showcased with a complete end-to-end work-flow analyzing a collaboration network of scientists from the field of complex networks. Finally, concluding remarks are given in Section 7.

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. First we review the estimation of A_k in isolation in Section 2.1, then the estimation of η_i in isolation in Section 2.2, and finally the joint estimation of A_k and η_i in Section 2.3. In the course of doing so, we also review the underlying temporal network models assumed in each case.

2.1. Attachment function estimation

The methods for estimating the attachment function A_k in isolation assume a simplified version of (1), in which the η_i are assumed to be 1. Thus the probability P_i in (1) only depends on A_k. Perhaps the most frequently-encountered parametric version of this model is the log-linear for $A_k = k^\alpha$ with attachment exponent $\alpha > 0$. 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 $\alpha = 1$ gives rise to a power-law distribution.

Concerning this model, there are three main estimation methods for 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 only holds when $\alpha = 1$, thus the method at best is heuristic 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 Minorize-Maximization algorithm (Hunter and Lange 2000). We defer the details of this term to Section 2.3. 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). The method is not only able to recover A_k accurately, but also can estimate confidence intervals of the estimated A_k for each k. It mains drawback is that it might be slow, since it is an iterative algorithm.

2.2. Node fitness estimation

When we consider only node fitnesses, there are two generative models in the literature with different assumptions on the functional form of A_k in (1). While the Caldarelli model (Caldarelli *et al.* 2002) assumes that A_k is 1 for all k, the BB model (Bianconni 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 Minorize-Maximization algorithms that maximize the corresponding log-likelihood functions with a regularization term that regularizes the distribution of η_i. 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 of the regularization 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 do not bother to include it in the package.

2.3. Joint estimation of the attachment function and node fitnesses

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

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 temporal network models that are special cases of model defined by (1).

The objective function of PAFit is a combination of the log-likelihood of the full model defined by (1) and two regularization terms: one for A_k and one for η_i. While we refer readers to Pham *et al.* (2016) for details on the log-likelihood function, we will discuss the regularization terms here.

The regularization term for A_k is defined by

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

with $w_k = \sum_{t=1}^{T} m_k(t)$ and $m_k(t)$ is 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 in log-scale. In the limit case when r is very large, we effectively assume that $A_k = k^\alpha$. Thus this covers the case of $\alpha = 1$ in the BA or the BB models and the case of $\alpha = 0$ in the growing ER or the Caldarelli model.

The regularization term for node fitnesses is defined by

$$\sum_i ((s-1) \log \eta_i - s\eta_i). \tag{3}$$

This term controls the variance of the distribution of node fitnesses. 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 effectively equivalent to estimate 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. As mentioned in the two previous sections, the values of r and s are automatically selected by cross-validation; see Table 2 in Pham *et al.* (2016) for a summary of the relation of r and s with well-known temporal network models.

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 η_i. The heavy computational parts of the package are implemented in C++ through the use of the **Rcpp** package (Eddelbuettel and François 2011; Eddelbuettel 2013). Furthermore, users with a multi-core machine 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 objects in the package. In what follows, we will review the main package functionalities one by one.

Name	Description
`generate_BA`	generates networks from the directed BA model
`generate_ER`	generates networks from the growing ER model
`generate_BB`	generates networks from the BB model
`generate_fit_only`	generate networks from the Caldarelli model
`generate_net`	generate networks with customizable PA function and node fitnesses
`get_statistics`	collects various statistics required in subsequent estimations
`only_A_estimate`	estimates the PA function in isolation
`only_F_estimate`	estimates node fitnesses in isolation with either $A_k = 1$ or $A_k = k$
`joint_estimate`	estimates the PA function and node fitnesses jointly
`coauthor`	contains the collaboration network dataset
`to_networkDynamic`	converts a `PAFit_net` object to a `networkDynamic` object
`from_networkDynamic`	converts a `networkDynamic` object to a `PAFit_net` object
`to_igraph`	converts a `PAFit_net` object to an `igraph` object
`from_igraph`	converts an `igraph` object to a `PAFit_net` object
`graph_to_file`	writes the graph in an `PAFit_net` object to file
`graph_from_file`	reads a graph from file into a `PAFit_net` object
`as.PAFit_net`	converts an edgelist matrix into a `PAFit_net` object

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

Firstly, most well-known temporal network models based on the 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 are tunable stochastic variables. They are actually wrappers of the more powerful `generate_net` function, which simulates networks with more flexible attachment function and node fitness settings. Given a PA function and a vector of node fitnesses, one can also use the function `simulate_true_net` to generate a network based on a true network as closely as possible. All statistics that are not relevant

to the PA and fitness mechanisms, e.g., the numbers of new nodes and new edges at each time-step, are kept the same as in the true network.

In any case, the output of these functions is a `PAFit_net` object, which is a list with four fields: `type`, `fitness`, `PA`, and `graph`. The `type` field is a string indicates 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 edgelist format in which raw temporal networks are stored in many online repositories (Kunegis 2013; Leskovec and Krevl 2014). We will discuss how to use functions provided by **PAFit** to convert this edgelist format to formats used in other network softwares in the next section. One can apply the function `plot` directly to a `PAFit_net` object to visualize its content.

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 subsequent estimation of PA and node fitnesses. The input network is assumed to be stored in a `PAFit_net` object. One can use the function `graph_from_file` to read an edgelist graph from a text file into a `PAFit_net` object, or convert an edgelist matrix to this class by the function `as.PAFit_net`.

The edgelist matrix is assumed to be in the same format as simulated graphs from **PAFit**, i.e., each row is of the form `(id_1 id_2 time_stamp)`. The node IDs are required to be integers bigger 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 is 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 time-stamps 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** is estimating 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 the 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 object of these functions contains the estimation results as well as some additional information pertaining to the estimation process. The estimated attachment function and/or node fitnesses can be plotted by using the `plot` command directly

on this output object. This will visualize not only the estimated results but also the remaining uncertainties when possible.

4. Related network analysis software

Since network analysis has been an important field for a long time, various aspects of it have been implemented in a very large number of software packages. To our best effort, we have confirmed that the estimations of PA and fitness mechanisms in a growing network are not implemented elsewhere. Restricting the discussion to packages in R, the main packages that deal with temporal networks are **igraph** (Csardi and Nepusz 2006), **statnet** (Handcock *et al.* 2008, 2016), and **RSiena** (Ripley *et al.* 2013).

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.

The **statnet** package is an extensive metapackage that contains many network-related **R** packages. Among them, packages that deal with temporal networks are: **networkDynamic** (Butts *et al.* 2016), **tsna** (Bender-deMoll and Morris 2016), and **tergm** (Krivitsky and Handcock 2016). 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 **tergm** package fits an exponential random graph model to a `networkDynamic` object. It allows very flexible modelling based on customizable network statistics. Although it theoretically allows the modelling of the PA function and node fitnesses as in our setting, pre-implemented functions in the package only allow for a simple parametric functional form of A_k. It also does not have regularization terms specifically designed for non-parametric A_k and η_i as in our package **PAFit**, which are essential for the joint estimation of the PA function and node fitnesses (Pham *et al.* 2016).

The **RSiena** package allows for versatile modeling of temporal networks using continuous-time exponential random graphs. It can accomodate discreate-time temporal networks, too. The package can model the probability of a new edge based on an extensive list of network statistics. It also implements various estimation methods, including the method of moments and likelihood-based methods. Like the case of **tergm**, it is theoretically possible to use **RSiena** to model the PA and fitness mechanisms. However, the pre-implemented functions in the package, as well as all the related usages of the package that we could find in the literature, have focused on estimating the simple parametric forms of the PA function $A_k = k$ and $A_k = \sqrt{k}$. Furthermore, like **tergm**, **RSiena** does not implement regularization terms that are crucial in joint estimation of the non-parametric PA function and fitnesses. Lastly, **RSiena** assumes that the node set is fixed in time, and thus cannot handle the addition of new nodes without introducing some approximations.

PAFit provides functionalities to communicate with these extensive 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 the directed version of the BA model, called Price's model (Price 1976). The network consists of $N = 1000$ nodes with $m = 5$ new edges added at each time-step.

```
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.

One can observe the emergence of hubs in this network by visualize the generated graph at various time-steps by the function `plot`. The following script plots the network snapshot at time $t = 10$ in Figure 1a and its corresponding degree distribution in Figure 1d:

```
R> plot(sim_net_1, slice = 1)
R> plot(sim_net_1, slice = 1, plot = "degree")
```

Note that if the network is directed, as it is in this example, the option `plot = "degree"` will plot the in-degree distribution. We can plot in the same way the 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 summary statistics of the temporal network:

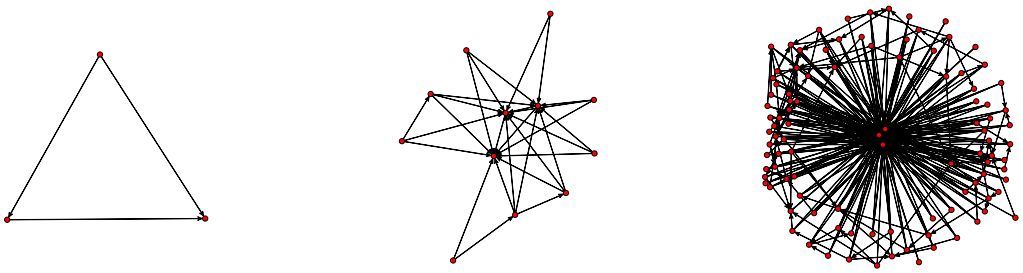
```
R> stats_1 <- get_statistics(sim_net_1)
```

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

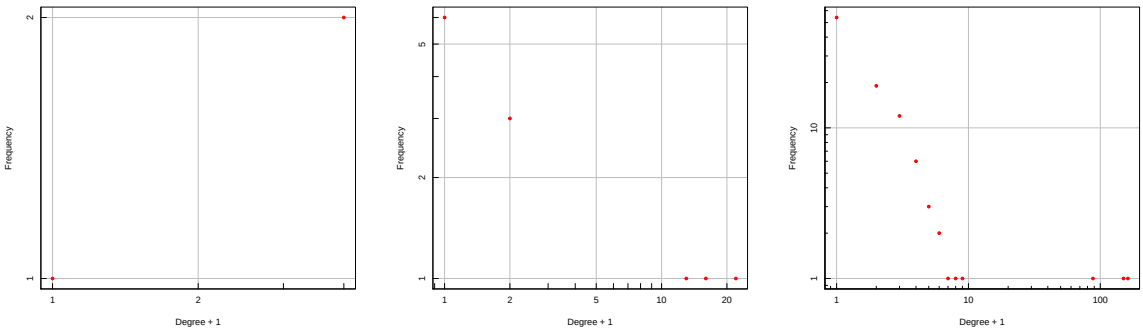
```
R> result_Jeong   <- Jeong(sim_net_1, stats_1)
R> result_Newman  <- Newman(sim_net_1, stats_1)
R> result_PA_only <- only_A_estimate(sim_net_1, stats_1)
```

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)
```



(a) Snapshot at $t = 1$ (b) Snapshot at $t = 10$ (c) Snapshot at $t = 100$



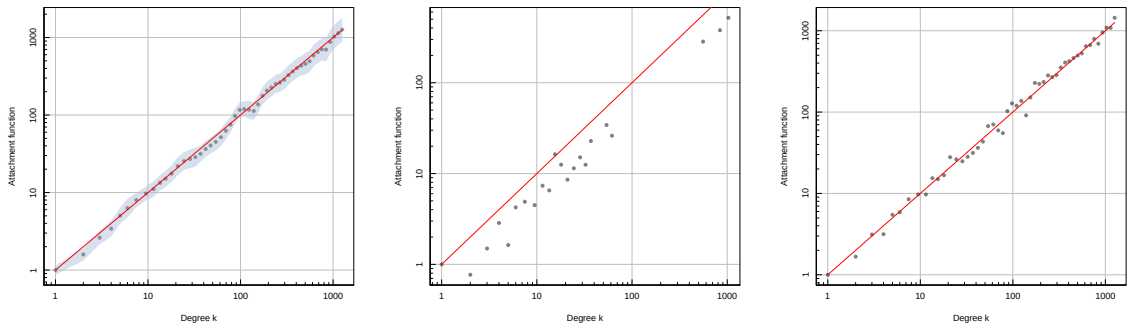
(d) Degree distribution at $t = 1$ (e) Degree distribution at $t = 10$ (f) Degree distribution at $t = 100$

Figure 1: Network snapshots and their corresponding in-degree distributions at time-steps $t = 1$, 10 and 100.

```
Estimation results by the PAFit method.
Mode: Only the attachment function was estimated.
Estimated r parameter:  0.1
Estimated attachment exponent:  1.001139
95% confidence interval of the attachment exponent: ( 0.9908913 , 1.011387 )
-----------------------------------------
Additional information:
Number of bins:  50
Number of iterations:  63
Stopping condition: 1e-08
```

As stated in Section 2, PAFit method first finds the r parameter, which regularizes the PA function, by cross-validation, and then estimate A_k using the chosen r. The estimated function can be assessed 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 two-sigma confidence interval are also estimated. Here $\hat{\alpha}$ is 1.001 ± 0.01 as we can see from the output of `summary`.

The output also reveals that in default **PAFit** applies binning with 50 bins. In this procedure, we divide the range of k into bins consist of consecutive degrees, and assume that all k in



(a) PAFit ($\hat{\alpha} = 1.001 \pm 0.01$)

(b) Jeong's method ($\hat{\alpha} = 0.96 \pm 0.07$)

(c) Newman's method ($\hat{\alpha} = 1.01 \pm 0.02$)

Figure 2: Estimating the attachment function in isolation. The true attachment function is $A_k = k^\alpha$ with attachment exponent $\alpha = 1$.

a bin have the same value of A_k. Binning is an important regularization 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 script, which produces Figure 2a.

```
R> plot(result_PA_only, stats_1)
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 recover the true α, while Jeong's method underestimates it. Note that in PAFit we also have the confidence intervals (lightblue region in Figure 2a) of the estimated A_k, which are unavailable in the other two methods. This is a significant advantage of PAFit over the two 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> 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")
```

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

In its default setting, the function `only_F_estimate` estimates node fitnesses under the assumption that $A_k = k$. But one also can estimate node fitnesses in the Caldarelli model, i.e., assuming $A_k = 1$ for all k, with the option `model_A = "Constant"`. The function `only_F_estimate` works by first find the estimated value $\hat{s}$ of s by cross-validation, and then uses $\hat{s}$ 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.
Estimated s parameter: 8
-----------------------------------------
Additional information:
Number of bins:  50
Number of iterations:  19
Stopping condition: 1e-08
```

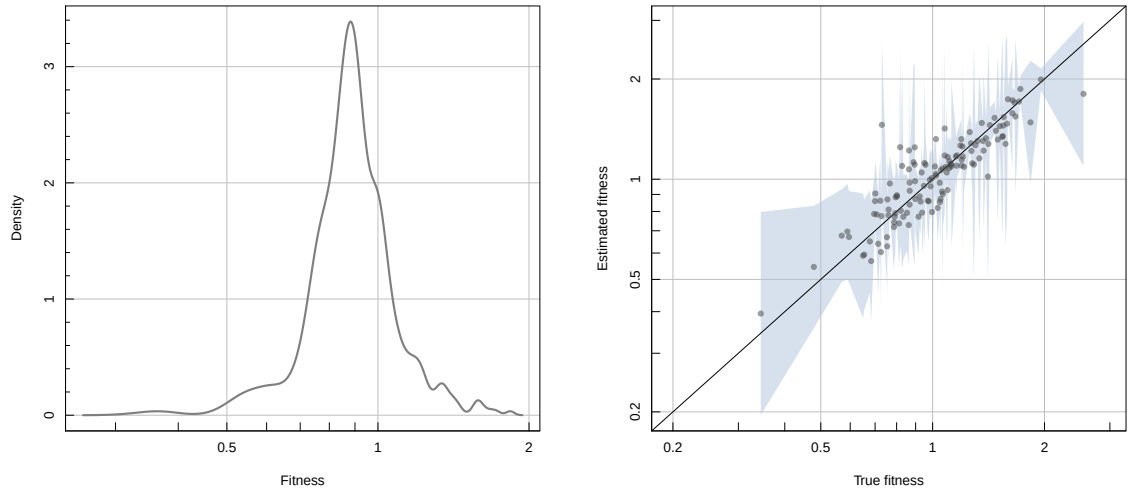
We can see that s is slightly underestimated, which means the variance of node fitnesses is overestimated. We can check whether the node fitnesses were estimated well by plotting the estimated fitnesses versus the true fitness by the following script:

```
R> plot(result_fit_only, stats_2, true_f = sim_net_2$fitness, plot = "true_f")
```

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 each $\hat{\eta}_i$ depicts the confidence intervals of that estimated values. 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 estimation of A_k in isolation, and in Section 5.2 a specific functional form of A_k for estimation of η_i in isolation. Such assumptions becomes unnecessary when we perform joint estimation, since the appropriate



(a) Distribution of estimated fitnesses. (b) Estimated fitnesses versus true fitnesses.

Figure 3: Estimating node fitnesses in isolation. The attachment function 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.

functional forms will be automatically enforced through 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.

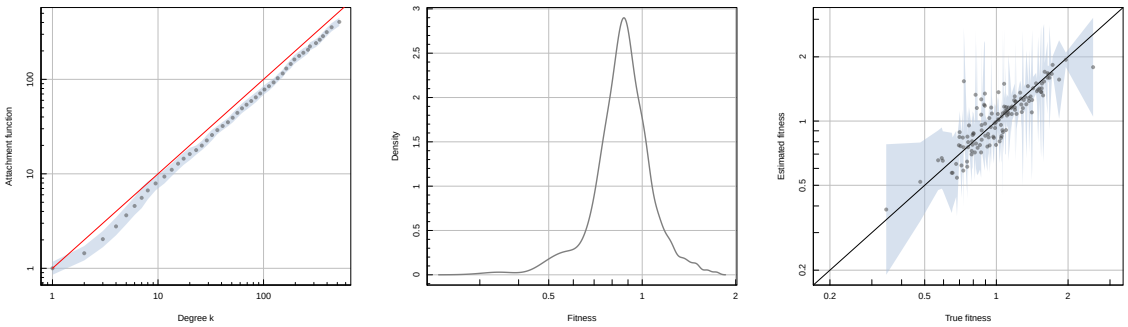
Using the same simulated network in Section 5.2, we apply `joint_estimation`:

```
R> result_PAFit <- joint_estimate(sim_net_2, stats_2)
R> summary(result_PAFit)

Estimation results by the PAFit method.
Mode: Both the attachment function and node fitness were estimated.
Estimated r parameter:  1.071875
Estimated s parameter: 7.5
Estimated attachment exponent:  0.9942476
95% confidence interval of the attachment exponent: ( 0.9836717 , 1.004824 )
-----------------------------------------
Additional information:
Number of bins:  50
Number of iterations:  399
Stopping condition: 1e-08
```

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

```
R> plot(result_PAFit, stats_2)
```



(a) Attachment function ($\hat{\alpha} = 0.99 \pm 0.01$)

(b) Distribution of estimated fitnesses

(c) Estimated and true η_i

Figure 4: Joint estimation of the attachment function and node fitnesses. The red line in the first panel is the true attachment function $A_k = k$.

```
R> lines(stats_2$center_k, stats_2$center_k, col = "red")
R> plot(result_PAFit, stats_2, true_f = sim_net_2$fitness, plot = "true_f")
```

Recalling that true α is 1, we can see that $\hat{\alpha} = 0.99 \pm 0.01$ is a good estimated. We can also plot the estimated fitnesses versus the true fitnesses as in Figure 4c with the following code:

```
R> plot(result_PAFit, stats_2, true_f = sim_net_2$fitness, plot = "true_f")
```

Although s is underestimated ($\hat{s} = 7.5$), the estimated fitnesses agree well with the true fitnesses.

6. Analysis of a collaboration network between scientists

In this section, we demonstrate the complete workflow 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 are scientists and an undirected edge exists between them if and only if they have jointly written a paper. Since it contains no duplicated edges between two scientists, the degree of a node represents the number of collaborators of that scientist. 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). 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 edgelist matrix `coauthor.net` to a `PAFit_net` object, and get the summarized 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 of the temporal network
Type of network:  undirected
Number of nodes in the final network:  1498
Number of edges in the final network:  5698
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 5698 collaborations among them.

The next step is to use `joint_estimate` for joint estimation:

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

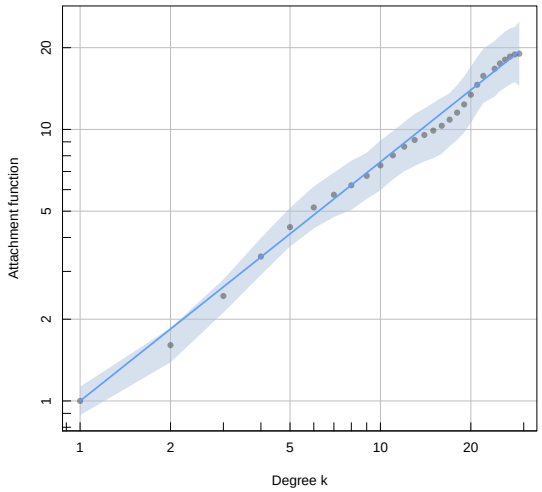
```
Estimation results by the PAFit method.
Mode: Both the attachment function and node fitness were estimated.
Estimated r parameter:  10
Estimated s parameter: 15
Estimated attachment exponent:  0.8803906
95% confidence interval of the attachment exponent: ( 0.8583292 , 0.902452 )
-----------------------------------------
Additional information:
Number of bins:  38
Number of iterations:  1200
Stopping condition: 1e-08
```

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

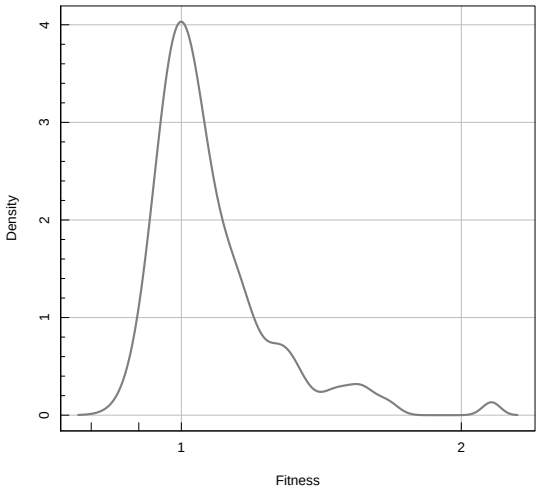
```
R> plot(full_result, net_stats, plot = "A")
R> plot(full_result, net_stats, plot = "f")
```

This snippet will sequentially generate Figures 5a and 5b.

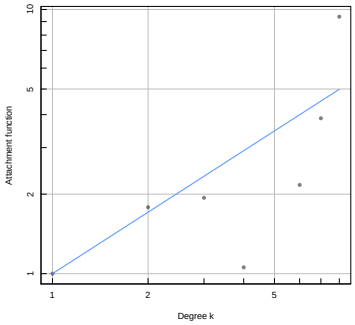
The best fit model when we performed joint estimation is close to the BB model. In Figure 5a, the estimated A_k is an increasing function on average with $\hat{\alpha} = 0.88 \pm 0.02$, which is close to 1. We can conclude that preferential attachment roughly exists in this collaboration network. Let us look at the region of small k, where the estimated attachment function is linear, for a concrete example: a network scientist with four collaborators has roughly twice the chance to get a new collaborator, compared with someone who only has two collaborators, assuming they have the same fitness. For comparison, we also plot the estimation results of A_k in isolation using Jeong's method, Newman's method and Pham *et al.* (2015)'s method in Figures 5c, 5d, and 5e, respectively:



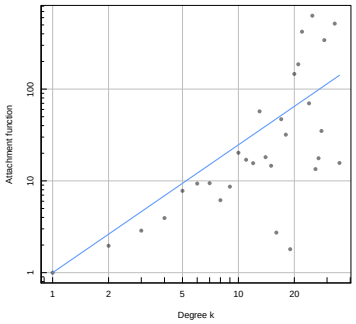
(a) Estimated A_k when joint estimation by PAFit ($\hat{\alpha} = 0.88 \pm 0.02$)



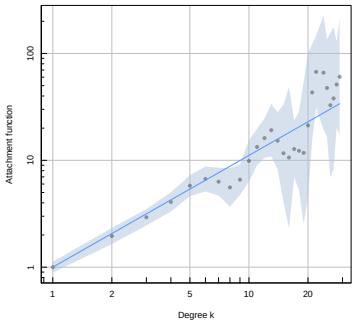
(b) Histogram of estimated node fitnesses



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



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



(e) Pham *et al.* (2015)'s method ($\hat{\alpha} = 1.05 \pm 0.07$)

Figure 5: Joint estimation of the attachment function and node fitnesses.

```
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, plot = "A")
R> plot(result_Newman, net_stats, plot = "A")
R> plot(result_onlyA, net_stats, plot = "A")
```

We notice that the estimated A_k of the joint estimation resembles that of Figure 5e, when we estimate it in isolation. The reason is that estimated node fitnesses in Figure 5b are highly concentrate 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 compared with that of Figure 5e. This is expected since in joint estimation, a portion of the connection probability in (1) is explained by node fitnesses.

Although the distribution in Figure 5b is concentrate 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 '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 snippet will produce the results show in Table 3. The table shows the top ten scientists that have the highest ability to attract new collaborators in the field of complex networks. If

Rank	Estimated fitness	Name
1	2.15	BARABASI, A
2	1.95	NEWMAN, M
3	1.73	JEONG, H
4	1.71	LATORA, V
5	1.66	ALON, U
6	1.66	OLTVAI, Z
7	1.63	WANG, B
8	1.61	YOUNG, M
9	1.60	SOLE, R
10	1.57	BOCCALETTI, S

Table 3: The top ten 'fittest' scientists in the field of complex networks.

one has some familiarity with the field, it is easy to recognize the names of many big-shots in the list. For example, at the top of the list is none other than 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.

7. Conclusion

We introduced the R package **PAFit**, which provides a comprehensive framework for the 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 summarized statistics from real-world temporal network datasets, and estimates 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. We believe that the package is useful for statistical analysis of temporal networks.

Acknowledgments

This work was supported in part by grants from the Japan Society for the Promotion of Science KAKENHI [JP16J03918 to T.P. and 16H01547 to H.S.].

References

Albert R, Barabási A (1999). "Emergence of Scaling in Random Networks." *Science*, **286**, 509–512.

Albert R, Jeong H, Barabasi AL (2000). "Error and Attack Tolerance of Complex Networks." *Nature*, **406**(6794), 378–382. URL http://dx.doi.org/10.1038/35019019.

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**, 69 – 77. ISSN 0378-4371. doi:http://doi.org/10.1016/S0378-4371(00)00018-2. URL http://www.sciencedirect.com/science/article/pii/S0378437100000182.

Bender-deMoll S, Morris M (2016). ***tsna**: Tools for Temporal Social Network Analysis*. R package version 0.2.0, 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. ISBN 1-59593-383-2. doi:10.1145/1143844.1143858. URL http://doi.acm.org/10.1145/1143844.1143858.

Bianconni G, Barabási A (2001). "Competition and Multiscaling in Evolving Networks." *Europhysics Letters*, **54**, 436.

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*.

Butts CT, Leslie-Cook A, Krivitsky PN, Bender-deMoll S (2016). ***networkDynamic**: Dynamic Extensions for Network Objects*. R package version 0.9.0, 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**, 258702. doi:10.1103/PhysRevLett.89.258702. URL http://link.aps.org/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**, 041902. doi:10.1103/PhysRevE.64.041902. URL http://link.aps.org/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. http://dx.doi.org/10.1137/070710111, URL http://dx.doi.org/10.1137/070710111.

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

Dorogovtsev SN, Mendes JFF (2003). *Evolution of Networks: From Biological Nets to the Internet and WWW (Physics)*. Oxford University Press, Inc., New York, NY, USA. ISBN 0198515901.

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*, **99**(20), 12917–12922. doi:10.1073/pnas.192407699. http://www.pnas.org/content/99/20/12917.full.pdf, URL http://www.pnas.org/content/99/20/12917.abstract.

Eddelbuettel D (2013). *Seamless R and C++ Integration with **Rcpp***. Springer-Verlag, New York. ISBN 978-1-4614-6867-7.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. URL http://www.jstatsoft.org/v40/i08/.

Eom YH, Jo HH (2014). "Generalized Friendship Paradox in Complex Networks: the Case of Scientific Collaboration." *Scientific Reports*, **4**. doi:10.1038/srep04603. URL http://dx.doi.org/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. http://dx.doi.org/10.1086/229693, URL http://dx.doi.org/10.1086/229693.

Gillespie CS (2015). "Fitting Heavy Tailed Distributions: The **poweRlaw** Package." *Journal of Statistical Software*, **64**(2), 1–16. URL http://www.jstatsoft.org/v64/i02/.

Guetz AN, Holmes SP (2011). "Adaptive Importance Sampling for Network Growth Models." *Annals of Operations Research*, **189**(1), 187–203. ISSN 1572-9338. doi:10.1007/s10479-010-0685-2. URL http://dx.doi.org/10.1007/s10479-010-0685-2.

Handcock MS, Hunter DR, Butts CT, Goodreau SM, Krivitsky PN, Bender-deMoll S, Morris M (2016). ***statnet**: Software Tools for the Statistical Analysis of Network Data*. The Statnet Project (http://www.statnet.org). R package version 2016.9, URL CRAN.R-project.org/package=statnet.

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. URL http://www.jstatsoft.org/v24/i01.

Hunter D, Lange K (2000). "Quantile Regression via an MM Algorithm." *Journal of Computational and Graphical Statistics*, pp. 60–77.

Jeong H, Néda Z, Barabási A (2003). "Measuring Preferential Attachment in Evolving Networks." *Europhysics Letters*, **61**(61), 567–572.

Kong J, Sarshar N, Roychowdhury V (2008). "Experience versus Talent Shapes the Structure of the Web." *Proceedings of the National Academy of Sciences of the USA*, **37**, 105.

Krapivsky P, Rodgers G, Redner S (2001). "Organization of Growing Networks." *Physical Review E*, p. 066123.

Krivitsky PN, Handcock MS (2016). ***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 3.4.0, URL http://CRAN.R-project.org/package=tergm.

Kunegis J (2013). "KONECT – The Koblenz Network Collection." konect.uni-koblenz.de. 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, NY, USA. ISBN 978-1-4503-1889-1. doi:10.1145/2464464.2464514. URL http://doi.acm.org/10.1145/2464464.2464514.

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. URL http://dx.doi.org/10.1038/35082140.

Momeni N, Rabbat MG (2015). *Measuring the Generalized Friendship Paradox in Networks with Quality-Dependent Connectivity*, pp. 45–55. Springer-Verlag International Publishing, Cham. ISBN 978-3-319-16112-9. doi:10.1007/978-3-319-16112-9_5. URL http://dx.doi.org/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. ISSN 0378-4371. doi:http://doi.org/10.1016/j.physa.2006.07.017. URL http://www.sciencedirect.com/science/article/pii/S0378437106008090.

Newman M (2001). "Clustering and Preferential Attachment in Growing Networks." *Physical Review E*, **64**(2), 025102.

Newman M (2006). "Finding Community Structure in Networks Using the Eigenvectors of Matrices." *Physical Review E*, **74**, 036104. doi:10.1103/PhysRevE.74.036104. URL https://link.aps.org/doi/10.1103/PhysRevE.74.036104.

Newman M (2010). *Networks: an Introduction*. Oxford University Press, Inc., New York, NY, USA. ISBN 0199206651, 9780199206650.

Newman M, Forrest S, Balthrop J (2002). "Email Networks and the Spread of Computer Viruses." *Physical Review E*, **66**, 035101. doi:10.1103/PhysRevE.66.035101. URL https://link.aps.org/doi/10.1103/PhysRevE.66.035101.

Pastor-Satorras R, Vespignani A (2001). "Epidemic Spreading in Scale-Free Networks." *Physical Review Letters*, **86**, 3200–3203. doi:10.1103/PhysRevLett.86.3200. URL https://link.aps.org/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. URL http://dx.doi.org/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**. doi:10.1038/srep32558. URL http://dx.doi.org/10.1038/srep32558.

Pham T, Sheridan P, Shimodaira H (2017). ***PAFit**: Generative Mechanism Estimation in Temporal Complex Networks*. R package version 1.0.0.0, URL cran.r-project.org/package=PAFit.

Price DdS (1976). "A General Theory of Bibliometric and other Cumulative Advantage Processes." *Journal of the American Society for Information Science*, **27**, 292–306.

Redner S (2005). "Citation Statistics from 110 Years of Physical Review." *Physics Today*, **58**(6), 49–54. doi:10.1063/1.1996475. physics/0506056.

Ripley R, Boitmanis K, Snijders TA (2013). ***RSiena**: Siena - Simulation Investigation for Empirical Network Analysis*. R package version 1.1-232, URL https://CRAN.R-project.org/package=RSiena.

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. URL http://dx.doi.org/10.1371%2Fjournal.pone.0013580.

Simon HA (1955). "On a Class of Skew Distribution Functions." *Biometrika*, **42**(3-4), 425–440. doi:10.1093/biomet/42.3-4.425. http://biomet.oxfordjournals.org/content/42/3-4/425.full.pdf+html, URL http://biomet.oxfordjournals.org/content/42/3-4/425.short.

Sinatra R, Wang D, Deville P, Song C, Barabási AL (2016). "Quantifying the Evolution of Individual Scientific Impact." *Science*, **354**(6312). ISSN 0036-8075. doi:10.1126/science.aaf5239. http://science.sciencemag.org/content/354/6312/aaf5239.full.pdf, URL http://science.sciencemag.org/content/354/6312/aaf5239.

Wang D, Song C, Barabási AL (2013). "Quantifying Long-Term Scientific Impact." *Science*, **342**(6154), 127–132. ISSN 0036-8075. doi:10.1126/science.1237825. http://science.sciencemag.org/content/342/6154/127.full.pdf, URL http://science.sciencemag.org/content/342/6154/127.

Affiliation:

Thong Pham
Mathematical Statistics Team
Generic Technology Group, Center for Advanced Intelligence Project, RIKEN

Nihonbashi 1-chome Mitsui Building, 15th floor, 1-4-1 Nihonbashi, Chuo-ku, Tokyo
E-mail: thong.pham@riken.jp