okaapi: an R package for generating social networks based on trait preferences

Josefine Bohr Brask[1,2,3,*], Lauren JN Brent[4], Delphine De Moor[4,5]

1. Copenhagen Center for Social Data Science (SODAS), University of Copenhagen, Copenhagen, Denmark
2. Section for Ecology and Evolution, Department of Biology, University of Copenhagen, Copenhagen, Denmark
3. Department of Applied Mathematics and Computer Science (DTU Compute), Technical University of Denmark (DTU), Kongens Lyngby, Denmark
4. Centre for Research in Animal Behaviour, University of Exeter, Exeter, UK
5. Department of Primate Behavior and Evolution, Max Planck Institute for Evolutionary Anthropology, Leipzig, Germany

*Corresponding author. Email: bohrbrask@gmail.com

Abstract

Modelling of simulated networks with generative network models plays a central role for our understanding of the emergence and consequences of network structures. Accessible software that generates simulated networks based on relevant processes can facilitate the use of this important approach in behavioural ecology, and can help drive forward our understanding of animal social structures.

Here we present the R package 'okaapi'. This package can generate simulated networks based on a key driver of real social networks in many species, namely trait preferences (where individuals differentially socialize with others that have certain trait values, such as their sex, age, body size, etc.). The package provides tools for generating, visualising and quantifying trait preference networks. It uses a flexible modelling approach, where many different trait preferences can be modelled, and multiple trait preferences can affect the network simultaneously (as in real networks). It can both generate purely simulated networks, and networks based on trait data from real populations. The package can also be used for modelling networks with other node attribute effects than trait preferences, and may be useful not only for animal social networks, but also human social networks and non-social networks.

This paper provides an introduction to the okaapi package, including information on kinds of studies where the package may be useful, the content of the package, how to use the package, and examples of its use (with R code included). We hope that the okaapi R package will be useful in the field of behavioural ecology and other research areas, and that it will help facilitate the use of generative network modelling in the study of animal social systems and further the integration of this approach with empirical approaches.

Keywords: Generative network models, network modelling, network simulation, node attributes, social networks, social preferences, R package, trait preferences.

1. Introduction

Social networks are fundamental components of social systems, both in humans and non-human animals. Social networks are studied by two main approaches: empirical data analysis, and generative network modelling of simulated networks. In this paper, we introduce a new software package for generative network modelling, which creates simulated networks based on *trait preferences*, an important driver of real social network structures.

Generative network models (in the sense we consider them here) are algorithms that build artificial networks (Newman et al. 2018, Brask et al. 2025). In essence, a generative network model consists of rules for how network nodes link to each other, implemented as a computer code that can generate networks based on those rules. Generative network models have been used widely in network science and play a key role for our current understanding of the drivers and consequences of network structures (for an overview see Brask et al. 2025). For example, generative network models can be used to study how linking processes affect network properties (e.g. Barabási & Albert 1999, Caldarelli et al. 2002, Ilany & Akçay 2016, Cantor & Farine 2018, Brask et al. 2024), how structural network features affect network transmission and robustness (e.g. Moore & Newman 2000, Miller 2009, Sah et al. 2017, Romano et al. 2018, Evans et al. 2020, Cantor et al. 2021, Brask & Brask 2024), and how sampling biases affect inferred network structures (e.g. Franks et al. 2010, Silk et al. 2015, Farine & Strandburg-Peskin 2016; Weiss et al. 2021). While generative network models have been used in behavioural ecology to study animal social structures (reviewed in Brask et al. 2025), their use in this field is less prominent than in wider network science, and they may not currently be used to their full potential. An important prerequisite for their more widespread use is the availability of accessible software, which can generate simulated networks based on the generative processes that drive the real social structures (e.g. Silk & Gimenez 2023). One such process is *trait preferences*: behaviour where individuals differentially socialize with others that have certain trait values, such as their sex, age, body size, etc. (details in Section 3).

Social networks have been studied empirically in many species (Krause et al. 2015), and this research has revealed that traits, such as the age, sex and body size of individuals, play a key role for how individuals are connected in their social networks (reviewed in Brask et al. 2024). This prompted the development of a new generative network model, the *trait preference model* (Brask et al. 2024), which creates networks based on trait preferences. The model provides a flexible method for generating networks based on this key real-world generative process, where multiple traits can simultaneously affect the network via different preferences, as is the case in the real world (Brask et al. 2024). Network generation based on the trait preference model could be useful for studies of animal social networks in many species, as well as for human social networks, non-social networks and theoretical networks (details in next section). Therefore, we thought a software package with tools for using this model could be useful.

Here, we present the R package 'okaapi', which contains easy-to-use tools for generating, visualising and quantifying networks based on the trait preference model, along with detailed help pages and other accompanying resources. We hope that this package, together with other recent software (e.g. Silk & Gimenez 2023, Ross et al. 2024), can facilitate the use of generative network modelling in behavioural ecology, and can help drive forward our understanding of social and other networked systems.

In the following, we first consider for which kinds of studies the okaapi package may be useful. We then give a general introduction to the package, and describe its content. We thereafter explain how to use the package, and provide examples with R code available. We end with a brief conclusion.

2. Studies where the okaapi package may be useful

The okaapi package may be useful for research on a range of systems. Given the importance of trait preferences across species (Brask et al. 2024), the package could be used for research on social networks of many different species (including humans), and it may also be useful for research concerning various types of non-social networks, and unspecified networks in theoretical studies.

The package may also be used for different types of studies. The most obvious use may be in studies that concern trait preferences (for example studies investigating how different trait preferences affect network structure, as in Brask et al. 2024). The package may, however, also be useful in other types of studies. Firstly, it may be used in studies where node effects other than trait preferences are relevant, as the generative processes modelled by okaapi can alternatively interpreted as other processes than trait preferences (for example, if the size and geographical position of airports affect their chance of them being linked by a flight route, then the airport network could be modelled with okaapi). Secondly, it may be used in studies where network structures that okaapi can create are in themselves useful, regardless of the generative processes behind them (for example studies investigating the effect of structural network features on processes such as the transmission of disease and information).

While other R packages exist that include tools for modelling networks (e.g. igraph, Csárdi 2006), the okaapi package provides tools that focus specifically on trait preferences (or more broadly interpreted, node attribute effects), and uses a new, flexible approach for modelling this key generative process. The package therefore complements the existing tools.

Given the above, the okaapi package could potentially be useful for a range of research topics and questions, in behavioural ecology and other research areas.

3. Introduction to the okaapi package

The purpose of the okaapi package is to provide accessible tools for the generation and investigation of networks that are based on trait preferences (or node attribute effects more generally, Section 2). The name 'okaapi' stands for 'generation of networks based on social preferences for traits', and is inspired by the okapi (a rare and elusive mammal also known as the zebra giraffe; Mallon et al. 2015). The package uses the trait preference model (presented in Brask et al. 2024) to produce the networks. The okaapi package is currently available for R (see the Code availability statement for how to download it). In addition to the package, several accompanying resources are also available (see Table 1).

Table 1. An overview of currently available resources related to the okaapi R package.

Ressource	Description	Location
okaapi R package	Provides tools for network generation based on trait preferences, as well as network visualisation and network measurements.	See the Code availability statement in this paper.
okaapi function help pages	Provide detailed information about each function in the okaapi package, including function arguments (parameters), function output, and brief code examples demonstrating how to use the function.	Within the okaapi package (accessed the same way as any R function help page).
okaapi example R scripts	Provide code examples demonstrating how to use the okaapi package, which are more extensive than the code examples in the okaapi function help pages.	See the Code availability statement in this paper.
okaapi introduction paper	Provides a detailed introduction to the okaapi package.	In front of you.
Trait preference model paper	Presents the generative network model which the okaapi package uses for network generation (the trait preference model) and uses it to investigate the effect of trait preferences on network structure and function.	Brask et al. 2024 (see the reference list).

3.1. The trait preference framework used in okaapi

Before using the okaapi package, it is helpful to be familiar with the terminology and definitions for trait preferences used in the package. They follow that of the underlying generative network model: the trait preference model. Here we describe the essentials; further details and mathematical description of trait preferences and the trait preference model can be found in Brask et al. 2024.

3.1.1. Trait preferences

In okaapi, the term *trait preference* (also known as *trait-based social preference*, Brask et al. 2024) refers to behaviour where individuals differentially socialize with others that have certain trait values (such as a certain sex, age, body size, etc.). The term covers both active preferences, where individuals actively use the traits of others in their decisions of who to socialize with, and passive preferences, where the trait-based socializing is due to other factors than active choice (such as trait-dependent habitat preferences; see Brask et al. 2024 for details). Trait preferences are considered in this broad sense because active and passive preferences can be modelled equivalently with the trait preference model. A trait preference consists of a *preference type* combined with a *trait type.*

3.1.2. Preference types

The okaapi package uses a categorisation of preferences into two general types: 1) *similarity preferences,* where individuals' preference for others depends on how similar they are to themselves with regard to a trait, and 2) *popularity preferences*, where individuals prefer certain trait values regardless of their similarity to themselves. Each general type covers multiple similarity and popularity preference types. For example, similarity preferences cover preference for others that are similar to oneself (such as the same age), as well as preference for others that are *dis*similar, moderately similar, etc.; and popularity preferences cover preference for others that have high trait values (such as old age), as well as preference for others that have low trait values, average trait values, etc.

3.1.3. Trait types

In okaapi, a *trait type* refers to a distribution of trait values, which may be interpreted as any trait that fits with the distribution. A given trait type can thus be used to model multiple real-world traits. An example of a trait type could be a categorical trait with two categories, which could e.g. be used to model the sex of individuals.

3.1.4. Combining preference types and trait types to get trait preferences

To model a trait preference in okaapi, the user combines a preference type with a trait type. For example, preference for socializing with one's own sex may be modelled by combining a similarity preference with a categorical trait that has two categories. This gives a trait preference where individuals' preference for others depends on whether they are of the same sex.

3.2. The functionality of okaapi in a nutshell

The okaapi package generates networks where the way that individuals (network nodes) link to each other reflects trait preferences. The mechanisms of the underlying generative network model are described in detail in Brask et al. 2024. Briefly, it works by calculating a *social attraction value* for each pair of individuals based on their trait values and the preference types that are combined with each trait. The model then uses these social attraction values in the determination of which pairs of individuals will be socially connected and how strong their connections will be (i.e. the positions and weights of network links); pairs that have a higher social attraction are more likely to get a link, and the link is likely to be stronger.

The package can both create fully simulated networks based on settings chosen by the user (i.e. without using any empirical data), as well as simulated networks based on empirically measured trait values (such as sexes, ages, etc. from real populations; details in the following sections).

Key features of okaapi network modelling include the following: Firstly, it can model a range of different trait preferences: it includes both similarity and popularity preferences, and each of these can be combined with any type of trait (or other node attribute, Section 2). Secondly, it can generate networks based on no trait preference (i.e. random social linking), a single trait preference, or multiple simultaneously acting trait preferences, reflecting that several traits are often simultaneously of importance in real social networks (Brask et al. 2024). For example, individuals may both prefer to socialise with individuals of their own sex, and with older individuals, with their social connections being based on a combination of these preferences. Thirdly, the trait preferences may be of different importance, both in the sense that trait preferences acting simultaneously do not need to have the same importance (reflecting real-world networks where one trait may be more important in driving social connection than another trait), and in the sense that the trait preferences may overall have more or less importance compared to random linking. The okaapi package thus provides a flexible method for modelling trait preference networks, where the networks may be based on different preferences, which may act alone or together, and may have different importance.

3.3. Network structures generated by okaapi

Different trait preferences (and their combinations) give rise to different structures, and the okaapi package can therefore generate network structures with different structural characteristics such as modularity and centralization, with trait-based patterns such as trait assortment and correlations between trait values and individual social centrality, and with increases or decreases in network metrics such as clustering, path length, and degree variation (see Brask et al. 2024 for how specific trait preferences affect some of these structural characteristics). It can also create networks across the range from random to highly structured (because the preferences can have more or less importance compared to

random linking), and it can create networks where structural characteristics occur alone or together in different ratios (depending on the combination of trait preferences used). See Fig. 1 for examples of network structures created with okaapi (note, these are only a limited set of possible structures).

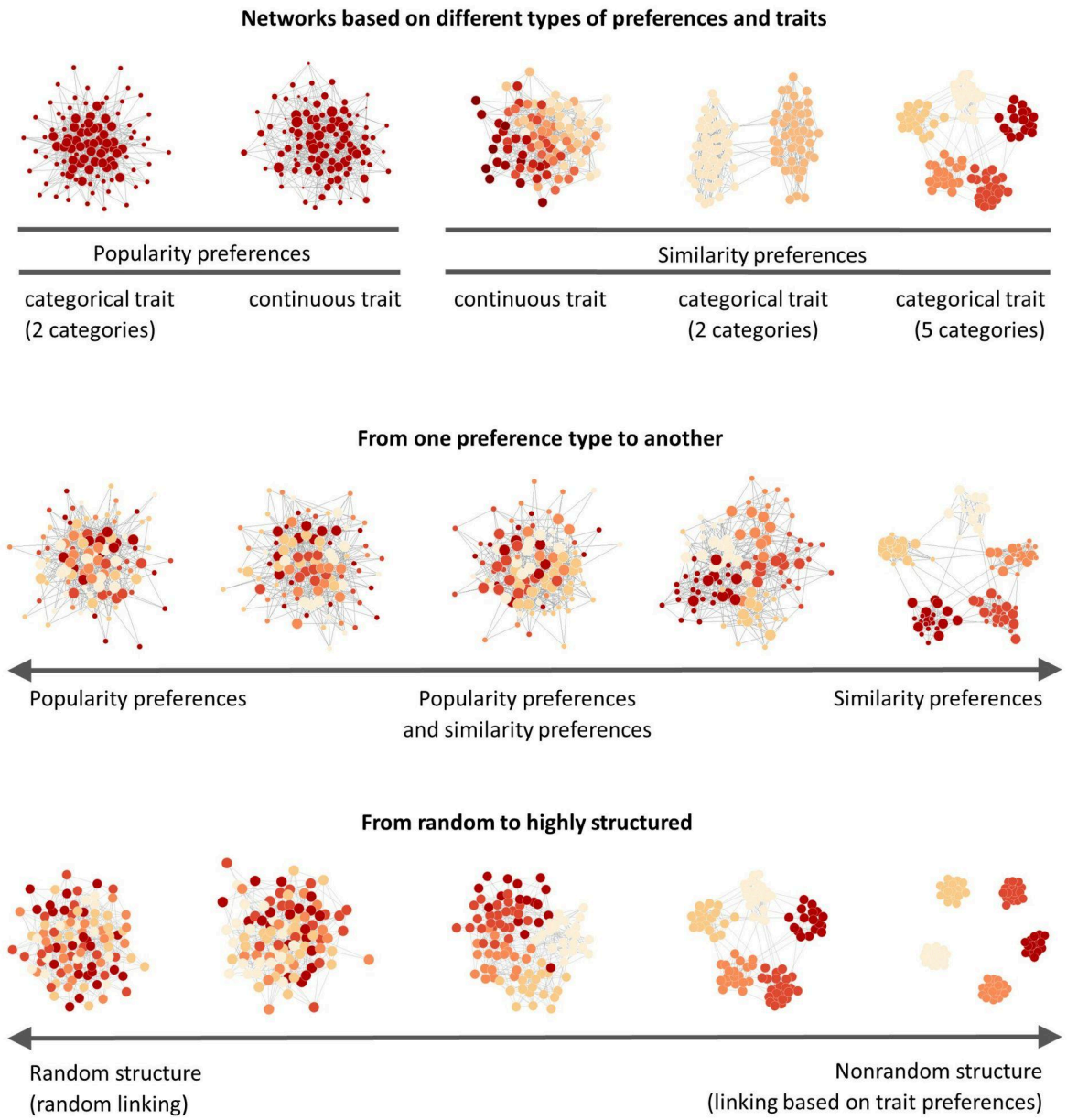

Figure 1. Examples of networks that can be generated with the okaapi package. Note that the package can create other network structures than those shown here, as the figure only includes networks based on a limited set of the possible trait types, preference types, and their combinations. The package can create networks based on different preference types, which can each be combined with different trait types (upper row); it can create networks based on a single or multiple trait preferences, which can be of different relative importance (middle row); and it can create networks from trait preferences that are overall of more or less importance compared to randomness (bottom row).

4. Contents of the okaapi package

Here we provide an overview and descriptions of the functions that the okaapi package contains. Detailed information about the arguments and output of each function is given in their respective help pages in R. See the next sections for how to use the functions.

4.1. Overview of okaapi functions

The current version of the okaapi package contains eight functions (listed in Table 2). There are two main functions: the *traitnet* function and the *traitnetsmetrics* function, which are respectively used to generate and plot a single network, and to quantify network metrics on an ensemble (a set) of networks. The remaining six functions are helper functions. The main purpose of these is that they are used by the two main functions (Fig. 2); but we have made them available as they may occasionally also be relevant to use as stand-alone functions.

Table 2. An overview of all the functions in the current version of the okaapi package, and their purposes. The helper functions are used by the main functions and may not be relevant to most users.

	Function name	**Function purpose**
Main functions	*traitnet*	Generate and plot a single network based on trait preferences
	traitnetsmetrics	Generate an ensemble of networks based on trait preferences and measure network metrics on them
Helper functions	*traitvalues*	Generate trait values (node attributes)
	traitnetsociat	Calculate social attraction values based on trait preferences
	traitnetbuild	Build a network matrix based on social attraction values
	contincols	Create network node colours based on values from a continuous scale
	traitnetvisual	Plot a trait preference network
	netmetrics	Measure network metrics on a network*
** This function only measures the metrics. To generate one or more networks based on trait preferences and measure metrics on it/them, use the traitnetsmetrics function*.		

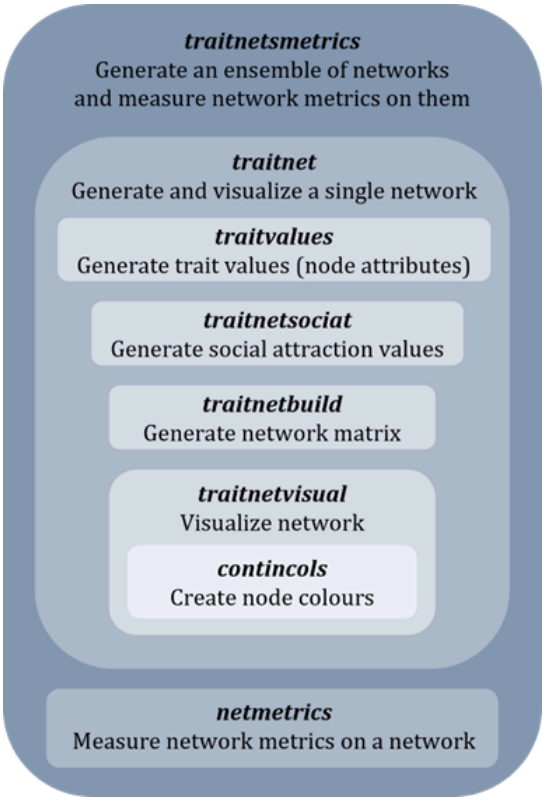

Figure 2. Structure of the okaapi package. The figure shows which okaapi functions use each other. Functions that are used by another function are nested within that function. For example, the *traitnetsmetrics* function calls the *traitnet* function and the *netmetrics* function. All functions are also available for users to use separately (but note that helper functions should be used with caution).

4.2. Description of okaapi functions

4.2.1. The traitnet function

This is the central function in the package, as it generates the network. It produces a single network based on user-specified settings, which it outputs as an adjacency matrix (i.e. a matrix containing the link weight for each pair of individuals). It also outputs the trait values of all individuals (nodes), the social attraction values of all pairs of individuals (see Section 3.2 and Brask et al. 2024), and (if desired) a visualisation of the network. The user can set a range of model parameters (which are given as arguments to the function), including the type(s) of trait(s), the type of preference used with each trait, and the importance of each trait preference (see Table 3 for overview of all the parameters). In addition to networks based on trait preferences, the function can also produce networks with random structure (if all preferences are set to zero importance). The *traitnet* function may be useful for example for visual exploration of networks based on different trait preferences, or for quantifying network features that are not included in the *traitnetsmetrics* function (see examples in Section 6).

Table 3. Parameter options for network generation with the okaapi package. For each parameter, the options that are available in the current version of the okaapi package are given and described, along with short names that are used in okaapi (note, option short names only exist for options where the argument input should be a string, and are therefore only given for these). The user gives the desired parameter options as function arguments to the *traitnet* and *traitnetsmetrics* functions. They include parameters for the trait preferences (parameter 1-4), for the network (parameter 5-9), and (when relevant) for user-provided traits (parameter 10-11). See Section 5 and the help files in R for further details about the parameters and parameter options (function arguments).

	Parameter	**Parameter short name used in okaapi**	**Parameter options**	**Parameter option short names used in okaapi**	**Description of parameter options**
1	**Trait types**	traittypes			
			categorical	'cate'	trait values are from a categorical distribution
			normal	'tnorm'	trait values are from a truncated normal distribution
			ranks	'ranks'	trait values are ranks (numbers with equal distance between them)
			user-provided	'own'	trait values are provided by the user
2	**Preference types**	preftypes			
			similarity	'sim'	individuals with trait values similar to one's own are preferred
			popularity	'pop'	individuals with high values of the trait are preferred
3	**Trait category numbers**	allncats			
			integer values or NA	-	the number of categories for categorical traits (NA for others)
4	**Importance weights**	wvals			
			values between 0 and 1	-	the importance of each trait preference
5	**Network size**	n			
			integer value	-	the number of individuals (nodes) in the network
6	**Average degree**	k			
			integer value smaller than the network size	-	the average number of links that each individual (node) has

					(i.e. this tunes the density of the network)
7	**Network link type**	linktype			
			stochastic weighted	'stow'	link weights are drawn from distributions with the social attraction values* as means
			deterministic weighted	'detw'	link weights are equal the social attraction values*
			unweighted	'unw'	links are unweighted (binary)
8	**Network components**	onecomp			
			one component	TRUE	the network consists of a single component
			one or more components	FALSE	the network can consist of one or more components
9	**Network visualisation**	visnet			
			network plot	TRUE	the network is visualised
			no network plot	FALSE	the network is not visualised
10	**Classes for user-provided traits**	owntraitclasses			
			no user-provided traits	NULL (default)	no trait values are given as input by the user
			categorical	'cate'	the user-provided trait will be treated as categorical
			continuous	'cont'	the user-provided trait will be treated as continuous
11	**Values for user-provided traits**	owntraitvals			
			no user-provided traits	NULL (default)	no trait values are given as input by the user
			for categorical traits: integers that indicate categories. For continuous traits: numbers between 0 and 1.	-	trait values for one or more user-provided trait(s)

** For explanation of social attraction values, see Section 3.2 and Brask et al. 2024.*

4.2.2. The traitnetsmetrics function

This is the second main function in the package. It quantifies network metrics (structural measures) on an ensemble of trait preference networks generated from a given set of parameters. This reflects a common approach in studies that involve simulated networks, where researchers work with a large set of networks of a given type (Brask et al. 2025): Generative network models are generally stochastic, in the sense that networks made from a given setting of model parameters are slightly different from one another (reflecting real-world random variation). This makes it possible to study distributions of network metrics from network ensembles rather than metrics from a single network, and thereby avoid biases. The networks in an ensemble may be considered as corresponding to replicates in an experimental treatment group. The *traitnetsmetrics* function uses the *traitnet* function to generate each network, and most of the parameters (arguments) are therefore the same as those of the *traitnet* function (Table 3). For the sake of efficiency, the network matrices are not retained; rather, the function generates and quantifies the networks sequentially, thus only requiring storage of a single network matrix at a time.

4.2.3. The helper functions

The remaining functions in the package (the helper functions, Table 2, Fig. 2) perform different parts of the network generation and metric quantification, and are used by the *traitnet* and *traitnetsmetrics* functions. These helper functions can also be used separately by the user (i.e. can be called directly), but they should in this case be used with caution due to requirements of their arguments, which must fit correctly together (for details see the R help pages of the respective functions). Of the helper functions, the *traitvalues*, *netmetrics*, and *contincols* functions may be particularly useful as stand-alone functions. The *traitvalues* function can be used to create sets of trait values (or more generally, node attributes or values drawn from different distributions). The *netmetrics* function can measure network metrics on a single input network (which does not have to be generated using the okaapi package). The *contincols* function can create colours from a continuous colour scale, which can be used to colour a network, and it is particularly useful for colouring networks visualised with the igraph R package (Csárdi 2006).

5. How to use the okaapi package

We here provide information on how to use the functions in the okaapi package. This is accompanied by a brief demonstration R code showing how to use the two main functions (see Code availability statement). More extensive examples (also with code) are described in the next section. Detailed explanation of the function arguments are given in Table 3 and in the function help pages in R.

5.1. Generating and quantifying networks using okaapi

To generate a network using the *traitnet* function, the user needs to provide it with arguments concerning the network (the network size, the average degree, the link type, whether the network should consists of a single component, and whether a plot of the network should be created), and arguments concerning the trait preference(s) that the network should be based on (for each trait preference: the trait type, the number of trait categories, the preference type, and the importance). If using user-provided trait values for some or all traits, the user also needs to input trait values and classes for those traits (see Section 5.3 and 6.4 below). When the user has set the arguments, they can run the function.

To quantify network metrics using the *traitnetsmetrics* function, the user needs to provide the same arguments as to the *traitnet* function, except for whether the network should be visualised (this is inconvenient for the purpose of *traitnetsmetrics* and therefore suppressed; a visualisation of the type of network that the metrics are measured on could be made by using the *traitnet* function with the same network generation settings). The user also needs to specify which network metrics should be computed, how many replicates (networks) they should be computed on (i.e. the size of the ensemble), and whether the current replicate number should be written to the console (this is to enable the user to follow the progression of the task, as this may take more than an instant, depending on the network size and number of replicates). The user can then run the function.

5.2. Setting function arguments correctly

To ensure that the input arguments the user has provided fit with the requirements, the functions contain a number of checks of the arguments, and will give a message if an error is found. However, some types of errors are impossible to check for. We therefore encourage users to do the following two things, which should make the risk of errors minimal:

1) Read the function help page and be sure that your arguments fit with the requirements.
2) Make sure that values within each argument are in the same order.

Regarding 1), the requirements of the arguments are described in detail in the function help pages. Regarding 2), when okaapi is used to generate or quantify networks based on more than one trait, then some arguments need to contain multiple values (one for each trait). For example, one argument will contain the trait type for each trait, another argument will contain the preference type used with each trait, a third will contain the importance of each trait, etc. For such arguments, the order of values in the arguments must be the same. For example, all the values at the first place of each argument must provide information that has to do with the same trait, all the ones on the second place must have to do with the next trait, etc. See examples in the function help pages and in the codes accompanying this paper.

5.3. Using simulated and real trait data

The trait values used in the network generation can either be generated by the *traitnet* or *traitnetsmetrics* function (based on the user's trait type settings), or given as input to the functions by the user. These two possibilities can be combined, such that values for some of the traits are function-generated and others are user-provided. User-provided trait values may be data measured on a real-world population, or simulated data that have been generated in some other way than with okaapi.

For function-generated traits, the user can select different trait types that are available in the package (Table 3). For each trait type that is set, a trait value for each network node (individual) will be drawn from the relevant distribution. The difference between the trait types lies in how their trait values are distributed (as explained in Section 3.1.3). To model a specific real-world trait, the user may choose a trait type with a trait value distribution that fits with that trait. For example, the user might set the trait type to 'categorical' to model the sex of individuals (note that the user also always needs to set the number of categories, which must be given as NA for non-categorical traits).

For user-provided traits, the user provides a trait value for each network node. These values may correspond to any trait (or other node attribute), but may need to be transformed before being given as input, to fit with the argument requirements (details in the function help pages).

6. Examples of using the okaapi package

Here we provide examples of how to use the okaapi package, with accompanying R code (see Code availability statement). Simpler demonstrations of how to use the functions are given the brief demonstration code and function help pages.

For the examples, we focus on two trait preferences: preference for socializing with one's own sex, and preference for socializing with individuals that have a large body size. We model the former by a similarity preference combined with a categorical trait with two categories, and we model the latter by a popularity preference combined with a continuous, normally distributed trait (Table 3). In the following, we refer to these two trait preferences as the *sex similarity preference* and the *size popularity preference*, respectively.

Using these trait preferences, we show how to explore trait preference networks visually, and how to measure network properties on ensembles of trait preference networks. For the latter, we demonstrate how to do this for metrics that are available in okaapi, as well as for metrics and other network measures that are not currently implemented in okaapi. Finally, while all the above-mentioned demonstrations use trait values generated by okaapi, we end by showing how to generate and measure networks using user-provided trait values.

Case studies with accompanying code that uses the okaapi package can furthermore be found in Brask et al. 2025 (includes studies of network emergence, network robustness and statistical power analysis).

6.1. Example: Visual exploration of trait preference networks

First, we want to explore what networks based on the sex similarity preference and the size popularity preference look like. To do this, we use the *traitnet* function to create and visualise trait preference networks. Our approach is to set the function arguments according to the two trait preferences, generate networks, and look at the output network plots. We do this for the two trait preferences separately as well as both together. We could do this by writing code for the function three times (once for the sex similarity preference, once for the size popularity preference, and once for both); however, we find that it is more convenient to include both trait preferences, and then set the importance (Table 3) of either trait preference to zero when we want to explore the effect of the other trait preference by itself; in this way, we only need to write the code once (see the R code for details).

While not all structural features are easily discernible by the human eye, the visual exploration gives us a general impression of the structures that arise from the two preferences (Fig. 3). The exploration suggests that: 1) the sex similarity preference leads to a modular structure with a module (network community) for each sex (Fig. 3a), 2) the size popularity preference leads to a centralized structure with individuals of large body size being central (Fig. 3e), and 3) both preferences acting together leads to networks with a mix of these two structural features (Fig. 3b-d).

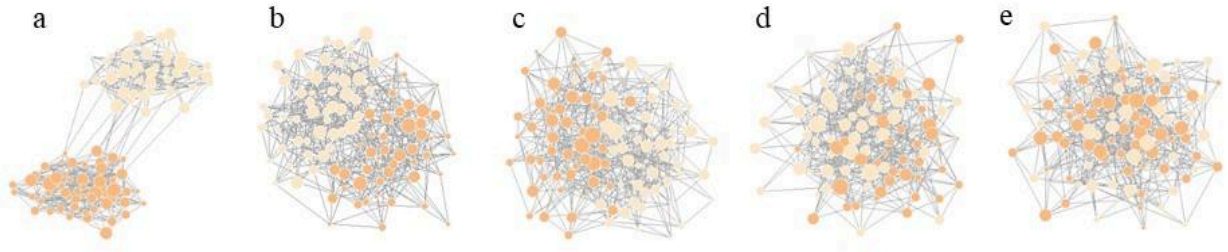

Figure 3. Visual exploration of networks based on the sex similarity preference and the size popularity preference. The figure shows networks generated with the *traitnet* function, based on the two preferences. Networks were generated based on either preference alone (a, e), or both together with varying relative importance (b-d). The colours correspond to sexes, and the node sizes correspond to body size. The importance weights used for these examples are as follows: a: sim 0.95, pop 0.00. b: sim 0.60, pop 0.30. c: sim 0.50, pop 0.50. d: sim 0.30, pop 0.60. e: sim 0.00, pop 0.95.

6.2. Example: Measuring structural aspects of trait preference networks with metrics that are available in okaapi

Next, we want to measure structural aspects of networks that are based on the two trait preferences. Reasons for doing this could be for example to quantify the effect of the preferences on network structure (as in Brask et al. 2024), or to investigate whether networks generated with these settings have desirable structural properties (e.g. for using them to study how structure affects spreading processes).

Our approach is to use the *traitnetsmetrics* function to generate large network ensembles and calculate metrics on them. We set the function arguments corresponding to the two trait preferences of interest (as above), and we also set arguments for the number of networks we want in the ensemble and the metrics we want to calculate (see R code for details). We then run the function and get the metric values for each network in the ensemble. We do this for four network ensembles: One for the sex similarity preference, one for the size popularity preference, one for both together, and one for no preferences, i.e. random networks (to have a baseline for comparison). For the ensembles with each preference alone, we choose to study the effect of the preferences when they have high importance, to see their effects most clearly. For the ensemble where they act simultaneously, we choose to study the case where the two preferences have equal importance, and we set the importance to the highest possible (which in this case is medium, because they have to share the importance). Similarly to above (Section 6.1), we find that instead of writing code for the function four times, it is more convenient to write it one time with both preferences included, and set the importance of either preference to zero when that preference should not be present (see R code for details).

We find that the sex similarity preference leads to networks with increased clustering and path length, and somewhat increased degree assortativity, compared to the networks with no preferences (Fig. 4). In contrast, the metrics from networks based on the size popularity preference are much more similar to those of the networks with no preferences. Interestingly, networks with both preferences have metrics that are much more similar to the networks based on the size popularity preference than those based on the sex similarity preference - despite the fact that we set the two preferences to have equal importance. This may be explained by the fact that the measured metrics increase non-linearly with the importance of the sex similarity preference (shown in Brask et al. 2024), with relatively little effect when its importance is medium (which it is in this example when the two preferences are acting together) compared to when its importance is high (which it is in the example when the preference is acting alone). Interestingly, in our visual exploration (Section 6.1 above), we saw that networks that were based on the two preferences with equal importance showed a clear pattern in terms of individuals of the same sex being closer in the network (Fig. 3c). This suggests that while the sex similarity preference with medium importance has limited effect on the three network metrics we measured here, it still has clear effects on the network, which could be quantified by measuring additional metrics (such as assortment by sex).

Here we used an approach where we made four network ensembles and compared them. An alternative to this would be to generate network ensembles across the whole range of importance for each trait preference (separately and together), and look at the metrics as a function of importance (as in Brask et al. 2024). The approach used here allows us to make easily interpretable box plots (Fig. 4); but it only tells us about the effect of the preferences for the specific importance settings we used for each ensemble. The other approach gives a fuller picture, as it shows the effect of the preferences across the full ranges of importance.

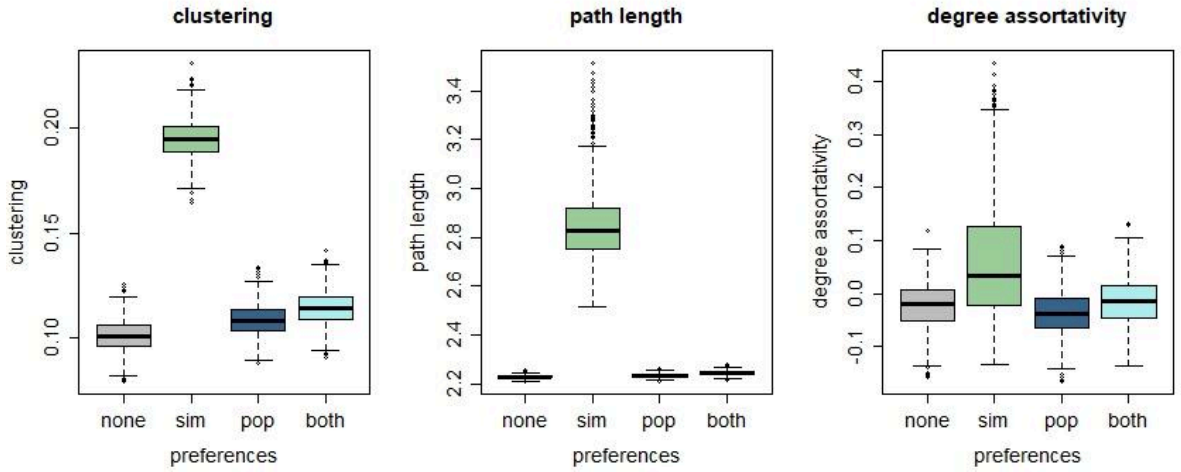

Figure 4. Measuring structural aspects of networks based on the sex similarity preference and the size popularity preference. The figure shows three network metrics (clustering, path length and degree assortativity) measured on four network ensembles: none = no preferences, sim = sex similarity preference, pop = size popularity preference, both = both of those preferences.

6.3. Example: Measuring aspects of trait preference networks with measures that are not available in okaapi

During our visual exploration, we noticed that the networks based on the different preferences differed in their modularity (the extent to which the network consists of distinct communities). We would therefore like to quantify this structural property. In the help page for the *traitnetsmetrics* function, we can see that modularity is currently not included as a metric in the function. It can, however, be computed by the help of other R packages. Instead of using the *traitnetsmetrics* function, we therefore make a code which creates networks with the *traitnet* function and measures their modularity with a function from another R package (igraph, Csárdi 2006). By using loops, we can do this for large network ensembles, similarly to the *traitnetsmetrics* function (see the R code for details).

We find that the modularity of networks based on the size popularity preference is almost indistinguishable from the modularity of the networks without preferences (random networks). In contrast, as we suspected from the visual exploration, the modularity is clearly higher for the networks based on the sex similarity preference than for the random networks, and this is to a smaller extent also the case for networks based on both preferences (Fig. 5).

We could use the same approach (combining the *traitnet* function with other code in loops) to investigate other properties of trait preference networks than metrics, such as their transmission efficiency and robustness.

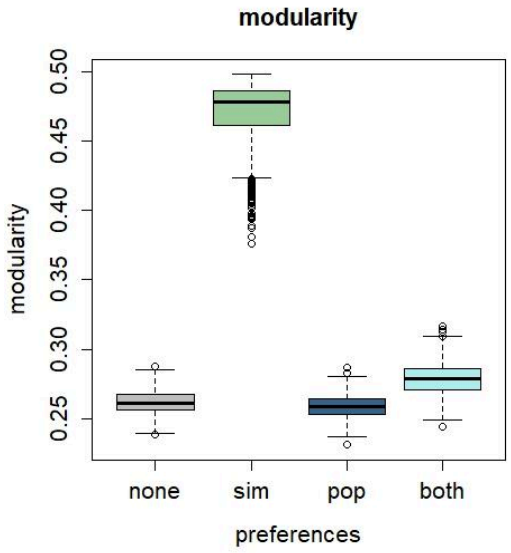

Figure 5. Measuring structural aspects of networks based on the sex similarity preference and the size popularity preference, using metrics that are not available in okaapi. The figure shows one network metric (modularity) measured on four network ensembles: none = no preferences, sim = sex similarity preference, pop = size popularity preference, both = both of those preferences.

6.4. Example: Measuring structural aspects of trait preference networks using user-provided trait values

We are here interested in a hypothetical, elusive study species. Preliminary observation has suggested that individuals of this species prefer to socialize with others of their own sex and who have a large body size (i.e. the two trait preferences used in the examples above). Luckily, a whole population of our elusive study species has just been spotted and their sexes and body sizes were recorded (we do not have data on their social network structure).

We can now use these observed trait data to investigate how the two presumed trait preferences may affect social structure, given the composition of individuals in the real population (i.e. the observed distributions of sex and body size trait values).

To do this, we use the *traitnetsmetrics* function in the same way as above (Section 6.2), except that instead of letting the function generate the trait values, we use our own trait values. To input the trait values, we use the *owntraitvals* argument (Table 3). We have two traits, and we therefore need to input a matrix that has one column for each trait and a row for each individual. We also need to set the class of each trait, using the *owntraitclasses* argument (Table 3). Finally, we set the network size so that it fits with the number of observed individuals. We use the same four ensembles, the same importance settings, and the same network metrics as above (Section 6.2).

The investigation gives us a base for understanding how social preferences observed in our species scale up to network properties, for the real trait value distributions. We find that the sex similarity preference has a strong influence on the network structure (all three metrics), whereas the size popularity preference has a much smaller effect. When both preferences are acting simultaneously (as the observations suggest that they do in our study population), then the networks are quite similar to random networks in terms of the three measured metrics. However, we should keep in mind that while we here let the two preferences share the importance equally when they were acting simultaneously, we do not know the relative importance of the two preferences in the real population. Given the strong effect of the sex similarity preference, the networks of the real population could potentially be farther from random networks, if that preference has higher importance in the real networks. Furthermore, our results only tell us about the three metrics that we have measured. Thus, next steps could be to investigate the effect of the two preferences across the range of their relative importance, and to measure more properties of the generated networks.

When comparing the results to those where we used trait values generated by okaapi (Fig. 6 versus Fig. 4), we see that for the networks with sex similarity preferences, degree assortativity is clearly higher when we used the observed trait values than when we used okaapi trait values. Thus, for the trait values of the observed population, the sex similarity preference leads to clearly increased degree assortativity, which was not the case when these preferences were combined with the okaapi trait values. This makes sense considering our observed trait data: compared to the okaapi trait values, the observed sex ratio was strongly skewed (whereas the observed body sizes were similar to the okaapi body sizes). When individuals then prefer to socialize with others of their own sex, the more numerous sex gets more connections, leading to increased assortment by degree. This demonstrates that using observed trait data can give additional insights about social networks for specific populations.

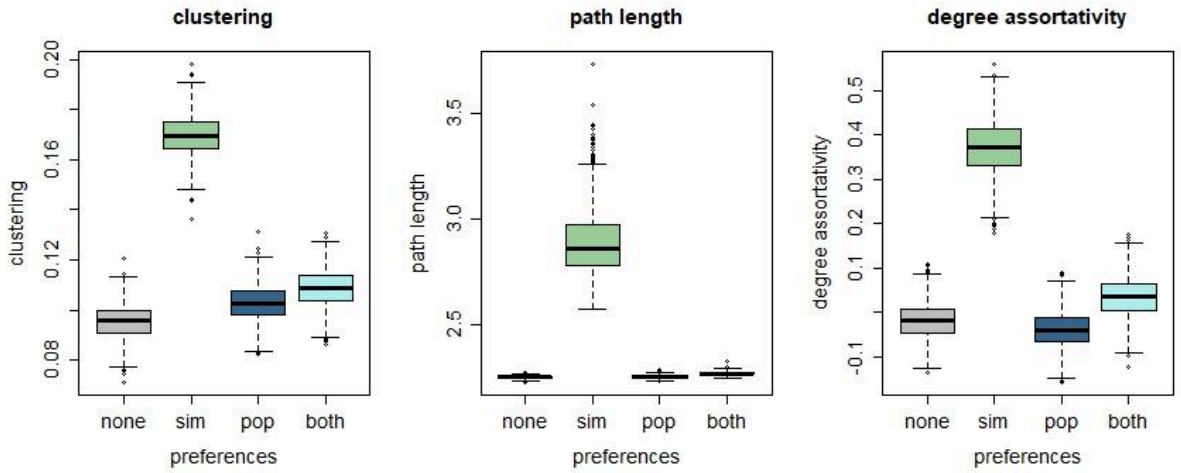

Figure 6. Measuring structural aspects of networks based on the sex similarity preference and the size popularity preference, using user-provided trait values. The figure shows three network metrics (clustering coefficient, path length and degree assortativity) measured on four network ensembles: none = no preferences, sim = sex similarity preference, pop = popularity size preference, both = both of those preferences. Here, the trait values used in the network generation were given as input rather than generated by okaapi.

7. Conclusion

Here we have presented the R package okaapi, which provides tools for generating, visualising and quantifying networks that are based on trait preferences (or equivalent node attribute effects).

In the future we may expand the functionalities of the package, for example by including additional similarity and popularity preference types, trait types and network metrics. We may also make an equivalent of the package for Python.

The okaapi package fills a gap by providing flexible network simulation tools that focus on trait preferences - a key driver of real social networks which is ubiquitously of importance across species - and it may also be used in studies that do not focus on trait preferences. We hope that the package can be useful for the behavioural ecology community and elsewhere, and that it can contribute to exciting new discoveries about networks and sociality.

Code availability

The currently available version of the okaapi package is the beta version, called okaapibeta. This will be updated to okaapi at a later stage. The okaapibeta package is available here: https://github.com/bohrbrask/okaapibeta. The package can be installed directly from within R. At the time of writing, this can be done by first installing the 'remotes' or 'devtools' package, and then running the following code: install_github("bohrbrask/okaapibeta").

R code for the brief demonstration and the examples can be found here: https://github.com/bohrbrask/okaapi-code-examples

See Table 1 for further information on where to find okaapi resources.

Author contributions

Josefine Bohr Brask conceived the idea of the okaapi package and developed it. The examples of how to use the package were developed by all authors and implemented by Josefine Bohr Brask. The writing of the manuscript was led by Josefine Bohr Brask, and all authors contributed significantly to the manuscript and approved the final version.

Acknowledgements

J.B.B. received funding from the Novo Nordisk Foundation (NNF23OC0085517) and the Carlsberg Foundation (CF20-0663). D.D.M. and L.J.N.B. acknowledge funding from a European Research Council Consolidator Grant awarded to L.J.N.B. (FriendOrigins—864461).

Conflict of interest statement

The authors declare no conflicts of interest.

References

Barabási, A. L., & Albert, R. (1999). Emergence of scaling in random networks. Science, 286(5439), 509-512.

Brask, J. B., Silk, M., & Weiss, M. N. (2025). An introduction to generative network models and how they may be used to study animal sociality. Animal Behaviour (in press).

Brask, J. B., Koher, A., Croft, D. P. & Lehmann, S. (2024). Far-reaching consequences of trait-based social preferences for the structure and function of animal social networks. arXiv, arXiv:2303.08107v2

Brask, J. B., & Brask, J. B. (2024). Evolution of cooperation in networks with well-connected cooperators. Network Science, 12(3), 305-320.

Caldarelli, G., Capocci, A., De Los Rios, P., & Munoz, M. A. (2002). Scale-free networks from varying vertex intrinsic fitness. Physical review letters, 89(25), 258702.

Cantor, M., & Farine, D. R. (2018). Simple foraging rules in competitive environments can generate socially structured populations. Ecology and evolution, 8(10), 4978-4991.

Cantor, M., Chimento, M., Smeele, S. Q., He, P., Papageorgiou, D., Aplin, L. M., & Farine, D. R. (2021). Social network architecture and the tempo of cumulative cultural evolution. Proceedings of the Royal Society B, 288(1946), 20203107.

Csárdi, G. & Nepusz, T. (2006) The igraph software package for complex network research. InterJournal Complex Systems, 1695.

Evans, J. C., Silk, M. J., Boogert, N. J., & Hodgson, D. J. (2020). Infected or informed? Social structure and the simultaneous transmission of information and infectious disease. Oikos, 129(9), 1271-1288.

Farine, D. R., & Strandburg-Peshkin, A. (2015). Estimating uncertainty and reliability of social network data using Bayesian inference. Royal Society open science, 2(9), 150367.

Franks, D. W., Ruxton, G. D., & James, R. (2010). Sampling animal association networks with the gambit of the group. Behavioral ecology and sociobiology, 64, 493-503.

Ilany, A., & Akçay, E. (2016a). Social inheritance can explain the structure of animal social networks. Nature communications, 7(1), 1-10.

Krause, J., James, R., Franks, D. W., & Croft, D. P. (Eds.). (2015). Animal social networks. Oxford University Press, USA.

Mallon, D., Kümpel, N., Quinn, A., Shurter, S., Lukas, J., Hart, J.A., Mapilanga, J., Beyers, R. & Maisels, F. 2015. Okapia johnstoni. The IUCN Red List of Threatened Species 2015: e.T15188A51140517.
https://dx.doi.org/10.2305/IUCN.UK.2015-4.RLTS.T15188A51140517.en.

Miller, J. C. (2009). Percolation and epidemics in random clustered networks. Physical Review E, 80(2), 020901.

Moore, C., & Newman, M. E. (2000). Epidemics and percolation in small-world networks. Physical Review E, 61(5), 5678.

Newman, M. (2018). Networks. Oxford university press.

Romano, V., Shen, M., Pansanel, J., MacIntosh, A. J., & Sueur, C. (2018). Social transmission in networks: global efficiency peaks with intermediate levels of modularity. Behavioral ecology and sociobiology, 72, 1-10.

Ross, C. T., McElreath, R., & Redhead, D. (2024). Modelling animal network data in R using STRAND. Journal of Animal Ecology, 93(3), 254-266.

Sah, P., Leu, S. T., Cross, P. C., Hudson, P. J., & Bansal, S. (2017). Unraveling the disease consequences and mechanisms of modular structure in animal social networks. Proceedings of the National Academy of Sciences, 114(16), 4165-4170.

Silk, M. J., Jackson, A. L., Croft, D. P., Colhoun, K., & Bearhop, S. (2015). The consequences of unidentifiable individuals for the analysis of an animal social network. Animal Behaviour, 104, 1-11.

Silk, M. J., & Gimenez, O. (2023). Generation and applications of simulated datasets to integrate social network and demographic analyses. Ecology and Evolution, 13(5), e9871.

Weiss, M. N., Franks, D. W., Brent, L. J., Ellis, S., Silk, M. J., & Croft, D. P. (2021a). Common datastream permutations of animal social network data are not appropriate for hypothesis testing using regression models. Methods in Ecology and Evolution, 12(2), 255-265.