AMERICAN PSYCHOLOGICAL ASSOCIATION

Psychological Methods

© 2021 American Psychological Association
ISSN: 1082-989X

https://doi.org/10.1037/met0000463

Semantic Network Analysis (SemNA): A Tutorial on Preprocessing, Estimating, and Analyzing Semantic Networks

Alexander P. Christensen[1] and Yoed N. Kenett[2]
[1] Department of Neurology, University of Pennsylvania
[2] Faculty of Industrial Engineering and Management, Technion Israel Institute of Technology

Abstract
To date, the application of semantic network methodologies to study cognitive processes in psychological phenomena has been limited in scope. One barrier to broader application is the lack of resources for researchers unfamiliar with the approach. Another barrier, for both the unfamiliar and knowledgeable researcher, is the tedious and laborious preprocessing of semantic data. We aim to minimize these barriers by offering a comprehensive semantic network analysis pipeline (preprocessing, estimating, and analyzing networks), and an associated R tutorial that uses a suite of R packages to accommodate the pipeline. Two of these packages, *SemNetDictionaries* and *SemNetCleaner*, promote an efficient, reproducible, and transparent approach to preprocessing linguistic data. The third package, *SemNeT*, provides methods and measures for estimating and statistically comparing semantic networks via a point-and-click graphical user interface. Using real-world data, we present a start-to-finish pipeline from raw data to semantic network analysis results. This article aims to provide resources for researchers, both the unfamiliar and knowledgeable, that reduce some of the barriers for conducting semantic network analysis.

Translational Abstract
We introduce a pipeline and associated tutorial for constructing semantic networks from verbal fluency data using open-source software packages in R. Verbal fluency is one of the most common neuropsychological measures for capturing memory retrieval processes. Our pipeline allows researchers who are unfamiliar or experienced with semantic network analysis to preprocess (e.g., spell-check, identify inappropriate responses), estimate, and analyze verbal fluency data as semantic networks, revealing the structural features of memory retrieval. The R packages support transparent and reproducible preprocessing of verbal fluency data, providing a standardized approach. Our empirical example demonstrates how these packages can be applied to real-world data. Finally, our pipeline is modular meaning different components (i.e., preprocessing, estimating, and analyzing) can be used in isolation, supporting other semantic tasks (e.g., free association, semantic similarity) as well as cross-software compatibility (e.g., Cytoscape, SNAFU in Python, other R packages).

Keywords: semantic networks, verbal fluency, semantic memory, network science

Supplemental materials: https://doi.org/10.1037/met0000463.supp

In recent years, network science has become an increasingly popular approach to study psychological phenomena such as language, learning, memory, personality, and psychopathology (Baronchelli et al., 2013; Borge-Holthoefer & Arenas, 2010; Borsboom & Cramer, 2013; Cramer et al., 2012; Fried & Cramer, 2017; Karuza et al., 2016; Siew et al., 2019). Network science is based on mathematical graph theory, which provides quantitative methods to represent and investigate complex systems as networks (Baronchelli et al., 2013; Siew et al., 2019). Network science methodologies provide a powerful computational approach for modeling cognitive structures such as semantic memory (i.e., memory of word meanings, categorizations of concepts and facts, and knowledge about the world; Kumar et al., 2021; Steyvers & Tenenbaum, 2005) and mental lexicon (i.e., word meaning, pronunciation, and syntactic characteristics; Stella et al., 2018; Wulff et al., 2019; for a review, see Siew et al., 2019).

In semantic network models, nodes (circles) represent semantic or lexical units, and edges (lines) represent the similarity, co-occurrence, or strength of the associations between them (Collins & Loftus, 1975). The nodes of a semantic network depend on the task; for example, they can be category exemplars (verbal fluency), associations to cue words (free association), or cue words whose similarities are rated (similarity

Alexander P. Christensen https://orcid.org/0000-0002-9798-7037
Yoed N. Kenett https://orcid.org/0000-0003-3872-7689

We are grateful to Cynthia Siew for her approval to include the *spreadr* package in the Shiny application and we thank her for the helpful feedback on the implementation. The authors did not preregister the study. All data, code, and materials can be found on the Open Science Framework: https://osf.io/hqxtc/

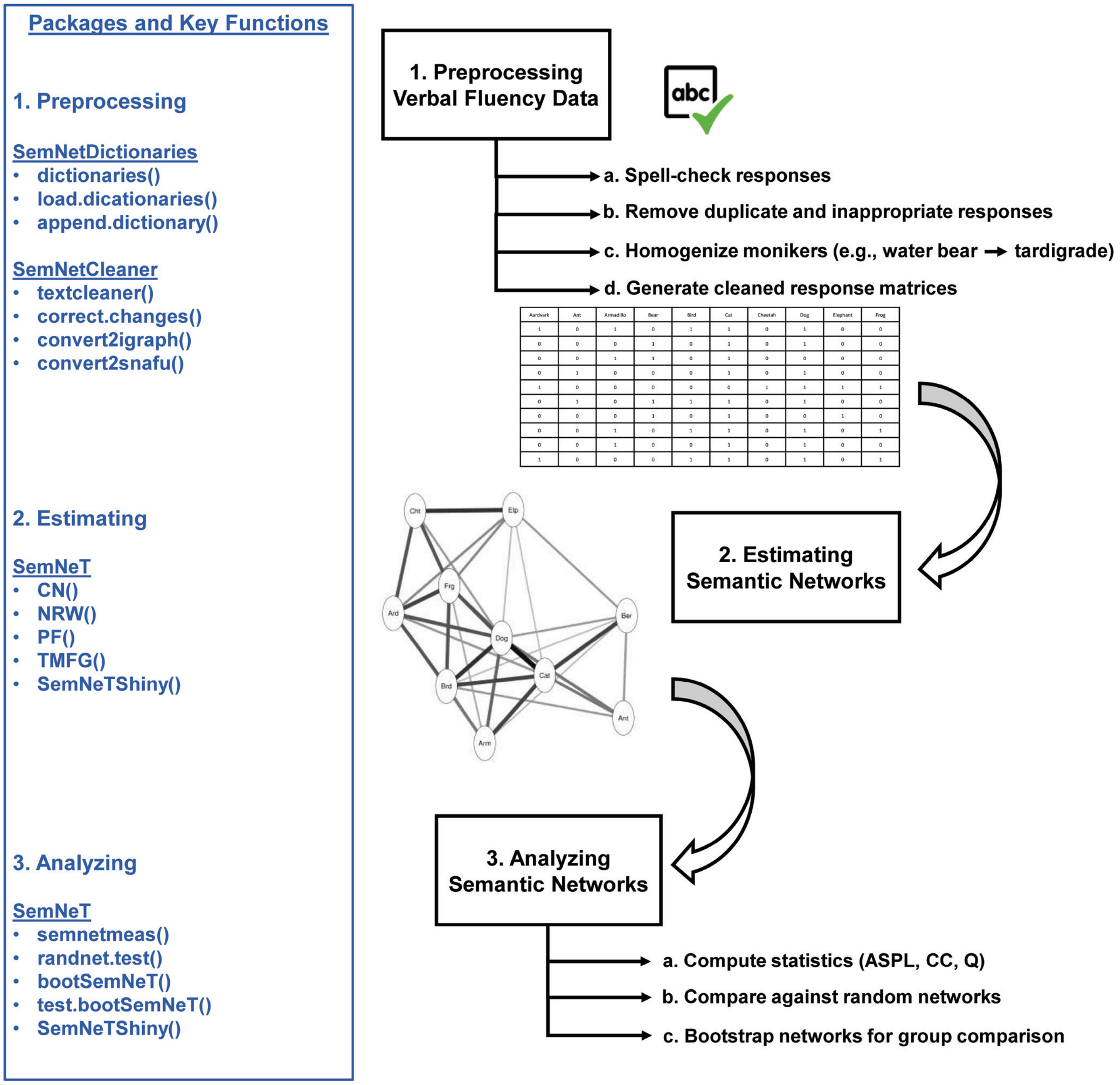
Correspondence concerning this article should be addressed to Alexander P. Christensen, Department of Neurology, University of Pennsylvania, 3700 Hamilton Walk, Philadelphia, PA 19104, United States. Email: alexpaulchristensen@gmail.com

judgments; De Deyne et al., 2016; Siew et al., 2019). For the purposes of the present article, we focus on semantic networks estimated from verbal fluency tasks; however, our pipeline is capable of handling other semantic memory tasks (e.g., free association and similarity judgments).

Despite the rise of network analysis in psychology, there are a couple of barriers that stand in the way of psychologists performing semantic network analysis (SemNA). One barrier is that there is a general lack of resources for how researchers can get started with conducting their own SemNA (Zemla et al., 2020). Another barrier is that preprocessing semantic data (e.g., spell-checking, removing inappropriate responses, making exemplars homogeneous, formatting the data for SemNA) is tedious and time-consuming.

The goal of this article is to minimize these barriers by introducing a suite of three R packages—*SemNetDictionaries*, *SemNetCleaner*, and *SemNeT*—that are designed to facilitate a comprehensive pipeline for preprocessing, estimating, and analyzing semantic networks (see Figure 1). This article is organized by integrating a detailed discussion on each step of the SemNA pipeline with associated R code that

Figure 1
A Step-by-Step Depiction of the Semantic Network Analysis (SemNA) Pipeline



Aardvark	Ant	Armadillo	Bear	Bird	Cat	Cheetah	Dog	Elephant	Frog
1	0	1	0	1	1	0	1	0	0
0	0	0	1	0	1	0	1	0	0
0	0	1	1	0	1	0	0	0	0
0	1	0	0	0	1	0	1	0	0
1	0	0	0	0	0	1	1	1	1
0	1	0	1	1	1	0	1	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	1	1	0	1	0	1
0	0	1	0	0	1	0	1	0	0
1	0	0	0	1	1	0	1	0	1

Note. ASPL = average shortest path length; CC = clustering coefficient; Q = maximum modularity coefficient; CN = community network; NRW = naïve random walk; TMFG = Triangulated Maximally Filtered Graph. See the online article for the color version of this figure.

provides a tutorial from raw data to statistical analyses. Our tutorial demonstrates how these R packages can be integrated into a single seamless pipeline; however, we emphasize that the pipeline can be modular such that researchers can perform one step following our tutorial (e.g., preprocessing) and then export their data to other R packages or software to perform different steps (e.g., network estimation and statistical analyses).

Semantic Networks Estimated From Verbal Fluency Data

One popular method to estimate semantic networks is via verbal fluency data (Siew et al., 2019; Zemla & Austerweil, 2018). Verbal fluency is a classic neuropsychological measure (Ardila et al., 2006) that asks participants to generate, in a short amount of time (e.g., 60 s), category members of a specific category. These categories can be semantic (e.g., animals) or phonological (e.g., words that start with "s"; Bousfield & Sedgewick, 1944). One strength of using verbal fluency tasks to study semantic memory is that they are quick to administer. In addition, some verbal fluency categories, such as animals, have a well-defined taxonomy that shows minor differences across different languages and cultures (e.g., animal kingdom; Ardila et al., 2006; Goñi et al., 2011).

To get from the raw verbal fluency data that participants provide (usually typed) to statistical analyses on semantic networks, three main steps are required: preprocessing the raw data, estimating networks from the preprocessed data, and statistically analyzing the networks (see Figure 1). Preprocessing the raw data includes spell-checking responses, identifying duplicate and inappropriate responses (commonly referred to as *perseverations* and *intrusions*, respectively), and homogenizing the naming scheme of exemplars into a common response (e.g., *water bear*, *moss piglets*, and *tardigrade* into *tardigrade*). Finally, networks are estimated from the preprocessed data using one of several network estimation approaches (Zemla & Austerweil, 2018). These estimated networks are then statistically analyzed and compared using different statistical approaches (e.g., random networks, bootstrap, random walks, spreading activation). Below, we discuss each step of the pipeline and provide associated R code.

Getting Started With the SemNA Pipeline in R

We used R Version 4.3 and RStudio Version 1.3.1093 to complete the pipeline. There are several R packages that need to be installed to setup the pipeline, which require R Version 3.6.0 or higher to be installed. The main packages are provided below (see SI1 for R session information):

```
# Install packages
install.packages(
  c("SemNetDictionaries", "SemNetCleaner",
    "SemNeT"),
  dependencies = c("Imports", "Suggests")
)
# Additional packages for Shiny GUI
install.packages(
  c(
    "shiny", "shinyjs", "shinyalert,"
    "shinyMatrix", "shinyBS"
  )
)
```

```
# Load packages
## SemNetCleaner automatically loads
  SemNetDictionaries
library(SemNetCleaner)
library(SemNeT)
```

Additional packages are also necessary for using the point-and-click graphical user interface (GUI) used in this tutorial. For Linux users, additional installations may be required for setting up Shiny server, which is necessary for the GUI (see https://rstudio.com/products/shiny/download-server/ubuntu/).

The reader can follow along with the article or use the tutorial R script found in the supplementary information (SI2). An additional R script template is also provided in the supplementary information (SI3) for readers interested in using this tutorial on their own data. Both the tutorial and template R scripts are also available on the Open Science Framework (https://osf.io/hqxtc/).

Our tutorial walks through the SemNA pipeline and its accompanying R packages using a verbal fluency dataset from one of our previously published papers (Christensen, Kenett, Cotter, et al., 2018). In this article, we examined a personality trait, openness to experience, and its relationship to semantic network structure. This dataset includes 516 participants who completed the animals verbal fluency task (1 min) and two measures of openness to experience: Big Five Aspects Scale (DeYoung et al., 2007) and NEO-FFI-3 (McCrae & Costa, 2007). The participants were evenly split into two groups based on low and high scores of openness to experience (both *N*'s = 258). The raw verbal fluency data is stored in the *SemNetCleaner* package and can be accessed using the following R code:

```
# Raw verbal fluency data from
 'SemNetCleaner'
data("open.animals")

# Structure of the data
head(open.animals)
```

The dataset is called open.animals, which is a data frame object that has 516 rows corresponding to each participant and 38 columns corresponding to variables. The first column is the group membership variable that denotes which group each participant belongs (1 = low openness to experience and 2 = high openness to experience). The second column contains each participant's openness to experience score. The third column is the participant's unique identifier variable (ID). The rest of the columns (4–38) contain each participant's animal names that they typed. Demographic and procedure details can be found in Christensen, Kenett, Cotter, et al. (2018).

Preprocessing Verbal Fluency Data

Regardless of how verbal fluency data is acquired, whether experimenter recorded or participant typed, several coding errors are likely to exist and need to be resolved before subsequent analyses. It's necessary, for example, to determine whether responses are perseverations (given by a participant more than once), intrusions (inappropriate exemplars for the category; e.g., animals category: *tasselled wobbegong*, *eucalyptus platypus*, *sugar bear*; a shark, small tree, and fictional character, respectively), or monikers (e.g., colloquial names or common misspellings) of a category exemplar (e.g., *water bear*, *moss piglet*, *tardigrade*). Other potential errors include typos, misspelled responses, inappropriate compound responses (i.e., responses where a space is missing between responses; e.g., *guineapig*), and response strings (i.e., multiple responses entered as a single string). After resolving these errors, the data still need to be processed into a format that can be used for network estimation. Moreover, commonly used behavioral measures such as perseverations and intrusions should be counted and extracted from the processed data.

To our knowledge, there is no standardized pipeline for preprocessing verbal fluency data meaning that different labs are likely making different decisions. This variability in decision-making has significant consequences that compound in the network estimation and statistical analysis steps. Recent work, where many research teams independently preprocessed or analyzed a single dataset, has demonstrated that the effects of these heterogeneous decisions can be a substantial issue for the reliability of results (Botvinik-Nezer et al., 2020; Silberzahn et al., 2018).

The lack of a standardized process and variation in analysis choices motivated the creation of two R packages called *SemNetDictionaries* and *SemNetCleaner*. The main purpose of these two packages is to automate and streamline the preprocessing of verbal fluency data while also minimizing preprocessing variability and the potential for human error. In addition, the output provided by the preprocessing step allows users to share their decisions to facilitate transparency and encourage open science practices (see SI4 and SI5). The *SemNetDictionaries* package assists *SemNetCleaner* by housing dictionary and moniker glossaries for several verbal fluency categories, while the *SemNetCleaner* package is the main preprocessing package. We first introduce the *SemNetDictionaries* package because it has several functions that are integrated into the *SemNetCleaner* package.

SemNetDictionaries Package

The *SemNetDictionaries* package contains several functions that facilitate the management and loading of dictionaries into *SemNetCleaner*. These dictionaries are used to spell-check and auto-correct the raw verbal fluency data in the preprocessing step. This package also includes a function that allows the user to create their own custom dictionaries, enabling them to save their own dictionaries for future use.

Predefined Dictionaries

The predefined dictionaries contained in the package currently include verbal fluency categories of *animals*, *fruits*, *vegetables*, and *jobs* as well as synonym categories of *hot* and *good* (see Table 1). A *general* dictionary also accommodates phonological fluency tasks with the use of single letters (e.g., words that start with "f") as well as more general text cleaning needs such as free association and semantic similarity tasks that are common in the semantic network literature (e.g., Nelson et al., 2004; Siew et al., 2019). For each category and synonym dictionary, there is an accompanying moniker glossary that is used to automatically convert monikers (e.g., *bear cat*) and common misspellings (e.g., *binterong*) into a homogeneous response (e.g., *binturong*). These categories were included because the authors had data for them; however, we note that there are other possible verbal fluency categories.

Table 1
Dictionaries in SemNetDictionaries

Dictionary	Entries	Task
Animals	1,210	Fluency (category)
Fruits	488	Fluency (category)
General	370,103	Fluency (category and phonological), free association, semantic similarity
Good	284	Fluency (synonym)
Hot	281	Fluency (synonym)
Jobs	1,471	Fluency (category)
Vegetables	284	Fluency (category)

The development of these category and synonym dictionaries included searching Google for lists of category exemplars and Thesaurus.com for synonyms. Their development was also aided by responses from several hundred participants, who generated verbal fluency responses for some of these categories (Christensen, Kenett, Cotter, et al., 2018) as well as responses from other published (Zemla & Austerweil, 2018) and unpublished data. Finally, the general dictionary was retrieved from the *dwyl* Github repository for English words: https://github.com/dwyl/english-words. Examples for how to load some of these dictionaries are provided below:

```
# Check for available dictionaries
dictionaries()
# Load 'animals' dictionary
load.dictionaries("animals")
# Load all words starting with 'f'
load.dictionaries("f")
# Load general dictionary
load.dictionaries("general")
# Load multiple dictionaries
load.dictionaries("fruits", "vegetables")
```

The load.dictionaries function loads as many dictionaries as are entered. The function alphabetically sorts and removes any duplicates found between the dictionaries. Thus, it returns an alphabetized vector the length of the unique words in the specified dictionaries.

Custom Dictionaries

A notable feature of the *SemNetDictionaries* package is that users can define their own dictionaries using the append.dictionary function. With this function, users can create their own custom dictionaries or append predefined dictionaries so that they can be used for future data cleaning and preprocessing. Researchers can create their own dictionary using the following code:

```
# Create a custom dictionary
append.dictionary(
  "your", "words", "here,"
  "in", "quotations", "and",
  "separated", "by", "commas,"
  dictionary.name = "example,"
  save.location = "choose"
)
```

All words that are entered in quotations and separated by commas will be input into a new custom dictionary. The name of the dictionary can be defined with the dictionary.name argument (e.g., dictionary.name = "example"). All dictionaries that are saved using this function have a specific file suffix (*.dictionary.rds), which allows the dictionaries to be found by the function find.dictionaries(). A user can also append a predefined dictionary (e.g., animals.

dictionary) by including the dictionary name in the function:

```
# Append a predefined dictionary
append.dictionary(
  animals.dictionary,
  "tasselled wobbegong,"
  dictionary.name = "new.animals,"
  save.location = "choose"
)
```

Appending a predefined dictionary does not overwrite it; instead, the user must save this dictionary somewhere on their computer (e.g., save.location ="choose"). Dictionaries from the *SemNetDictionaries* package and dictionaries stored using the append.dictionary function can be integrated into the *SemNetCleaner* package for preprocessing (we describe how this is done in the next section).

Moniker Glossaries

To view the moniker glossaries, the user must directly access the data file included in *SemNetDictionaries*. This can be done using the following code:

```
# Load 'animals' monikers
load.monikers("animals")
```

Similar to the dictionary files, all monikers included in the package will have a *.moniker suffix. Unlike the dictionaries, users are unable to save and load their monikers for future use in the *SemNetCleaner* package.

Crowdsourcing Dictionary and Moniker Glossaries

The dictionary and moniker glossaries are an expansive but not exhaustive resource for spell-checking for researchers. For the dictionaries included in *SemNetDictionaries*, the animals dictionary is the most developed. The animals dictionary is derived from several data sets and web scraping searches. All other dictionaries are derived from a few data sets. Because all dictionaries are nonexhaustive, participants will inevitably generate category exemplars that are not included in our dictionaries. Further, other categories exist that are not included in *SemNetDictionaries*. In these cases, the general dictionary can be used as a starter dictionary for preprocessing some of the data; however, users may wish to develop their own dictionaries.

Users are encouraged to submit their dictionary and moniker glossaries to our GitHub page: https://github.com/AlexChristensen/SemNetDictionaries/issues/new/choose. Our GitHub will serve as a centralized crowdsourcing database where dictionary and moniker glossaries can continually be updated. Dictionary files can either be created using the append.dictionary function in *SemNetDictionaries* or at the end of the preprocessing stage (as we detail below). Similarly, moniker glossaries can be created at the end of the preprocessing stage. Details for submitting these files are printed to the user at the end of the preprocessing and on the GitHub page. In the next section, we explain the preprocessing stage and the output that can be submitted to our GitHub.

SemNetCleaner Package

The *SemNetCleaner* package houses several functions for the preprocessing of semantic data. The purpose of this package is to facilitate efficient and reproducible preprocessing of verbal fluency data. Notably, other R packages perform similar functions (e.g., spell-checking, text mining) such as *hunspell* (Ooms, 2018), *qdap* (Rinker, 2020), and *tm* (Feinerer et al., 2008). However, the *SemNetCleaner* package sets itself apart from these other packages by focusing on commonly used tasks for SemNA (e.g., verbal fluency), which facilitates automated preprocessing.

The *SemNetCleaner* package applies several steps to preprocess raw verbal fluency data so that it is ready to be used for estimating semantic networks. These steps include spell-checking, verifying the accuracy of the spell-check, homogenizing responses, and obtaining a cleaned response matrix for network estimation. To initialize this process, the following code must be run:

```
# Run 'textcleaner'
clean <- textcleaner(data = open.animals
[,-c(1:2)], miss = 99,
       partBY  =  "row",  dictionary  =
"animals")
```

In the above code, we are removing the first two columns (group and openness to experience score variables) with the code [,-c(1,2)] because they are not relevant for our analysis. The only variables that should be entered into textcleaner are an optional column for participant's IDs and their respective verbal fluency data. The input data for textcleaner should have participants' response across the rows (or columns). An optional row (or column) can contain participant IDs. An example is provided in Table 2.

textcleaner is the main function that handles the data preprocessing in *SemNetCleaner* (for argument descriptions, see Table 3). For input into data, it's strongly recommended that the user input the full verbal fluency dataset and not data already separated into groups. If verbal fluency responses are already separated, then they will need to be input and preprocessed separately. Therefore, it's preferable to separate the preprocessed data into groups at a later stage of the SemNA pipeline.

Two arguments in the textcleaner function pertain to importing custom dictionaries created using the append.dictionary function in *SemNetDictionaries*. The dictionary argument allows

Table 2
Example of Data Input Into Textcleaner

Id	vf_an_01	vf_an_02	vf_an_03	vf_an_04	vf_an_05	vf_an_06
4	cat	dog	giraffe	elephant	monkey	chicken
6	cat	mouse	dog	raccoon	squirrel	snake
7	monkeys	cats	dogs	gorillas	zebra	fish
9	jaguar	cat	dog	mice	deer	dolphin
10	cat	dog	fish	bird	monkey	elephant
11	dog	cat	spider	cow	horse	pig

Table 3
Textcleaner Arguments

Argument	Description
data	A matrix or data frame object that contains the participants' IDs and semantic data.
miss	A number or character that corresponds to the symbol used for missing data. Defaults to 99.
partBY	Specifies whether participants are across the rows ("row") or down the columns ("col").
dictionary	Specifies which dictionaries from *SemNetDictionaries* should be used (more than one is possible). If no dictionary is chosen, then the "general" dictionary is used.
spelling	Changes the spelling of English output to either "UK" or "US" spelling. Defaults to "US."
add.path	The path to a directory where the user's saved dictionaries are stored. Allows for an expedited search for user created dictionaries. Defaults to NULL.
keepStrings	Whether strings (more than one word) should be kept together. Allows spelling checking of sentences. Defaults to FALSE.
allowPunctuations	Whether punctuations are allowed to be included in responses. Default allows dashes ("-") only. Input can include other punctuations (e.g., c("-", ".", ",")). "All" allows all punctuations.
allowNumbers	Whether numbers are allowed to be included in responses. Defaults to FALSE.
lowercase	Whether letters in responses should be all lowercase. Defaults to TRUE.
continue	A list object from a previous session of textcleaner. Allows progress to be continued after encountering an error or stopping the preprocessing step. Defaults to NULL.

the user to type the name of one or more dictionaries to be used for spell-checking including the names of dictionaries the user has created. Dictionary names entered into the dictionary argument that are in the *SemNetDictionaries* package will be found immediately while dictionaries that were created using the append.dictionary function are searched for on the user's computer. This search finds any files with the *.dictionary suffix in R's environment, temporary files, and current working directory as well as the user's desktop and downloads folders. Users can also specify a path to a directory where dictionaries are saved using the add.path argument. This latter argument is most useful when storing custom dictionaries in a single directory (e.g., a project's directory). Users can use add.path = "choose" to open an interactive directory explorer to navigate to a folder where there are saved dictionaries.

Spell-Check

By running the above code, textcleaner will start preprocessing the data immediately. The first step of textcleaner is to spell-check and attempt to auto-correct all responses. The processes of the automated spell-check are printed to the user (see Figure 2). Textcleaner first attempts to detect whether there is a unique identifier variable (ID) that was included in the data. The second message to print in Figure 2, IDs refer to variable: "ID," tells the user what variable the IDs in the output will refer to. In our example data, there was a variable named ID. If an ID variable is not provided, then the IDs in the output will refer to the row numbers (and the message will notify the user that this is the case). Next, the dictionary is converted into either United States ("US") or United Kingdom ("UK") word spelling (e.g., *color* or *color*, *theater* or *theater*, *gray* or *gray*, *program* or *program*, respectively).

The next five messages inform the user about the automated spell-checking processes. The first process separates the correctly spelled responses that were identified in the dictionary from responses not found in the dictionary. The second process checks for responses that are correctly spelled but are plural forms (e.g., *cats*) and converts them to their singular forms (e.g., *cat*). The third process checks for common misspellings and colloquial names and converts them into homogeneous responses. The fourth process attempts to automatically correct each remaining response (270 responses in our example) by individually evaluating the similarity between each response and responses in the dictionary using two edit distances or measures of (dis)similarity: Damerau-Levenshtein distance (DL; Damerau, 1964; Levenshtein, 1966) and QWERTY distance (or the sum of the physical distance—number of keys—between any two keys on a QWERTY keyboard). Finally, the fifth process attempts to parse responses that are more than one word (e.g., *guinea pig*). This last process may also separate responses that were accidentally typed as a single response by the participant as we show next (see Figure 3).

By pressing ENTER, the user can proceed to the manual spell-check where responses that were not able to be automatically corrected are checked by the user (in our example, only 38 need to be checked; Figure 2). The manual spell-check begins with more complicated cases first: responses where participants did not separate their responses—that is, when participants were typing their responses and did not hit a key (e.g., ENTER) to separate their responses, producing a string of multiple responses as if they were a single response. After these response strings are handled, then individual responses are checked. The manual spell-check uses an interactive menu where the user has complete control over how to correct responses. The first example of this menu is shown in Figure 3.

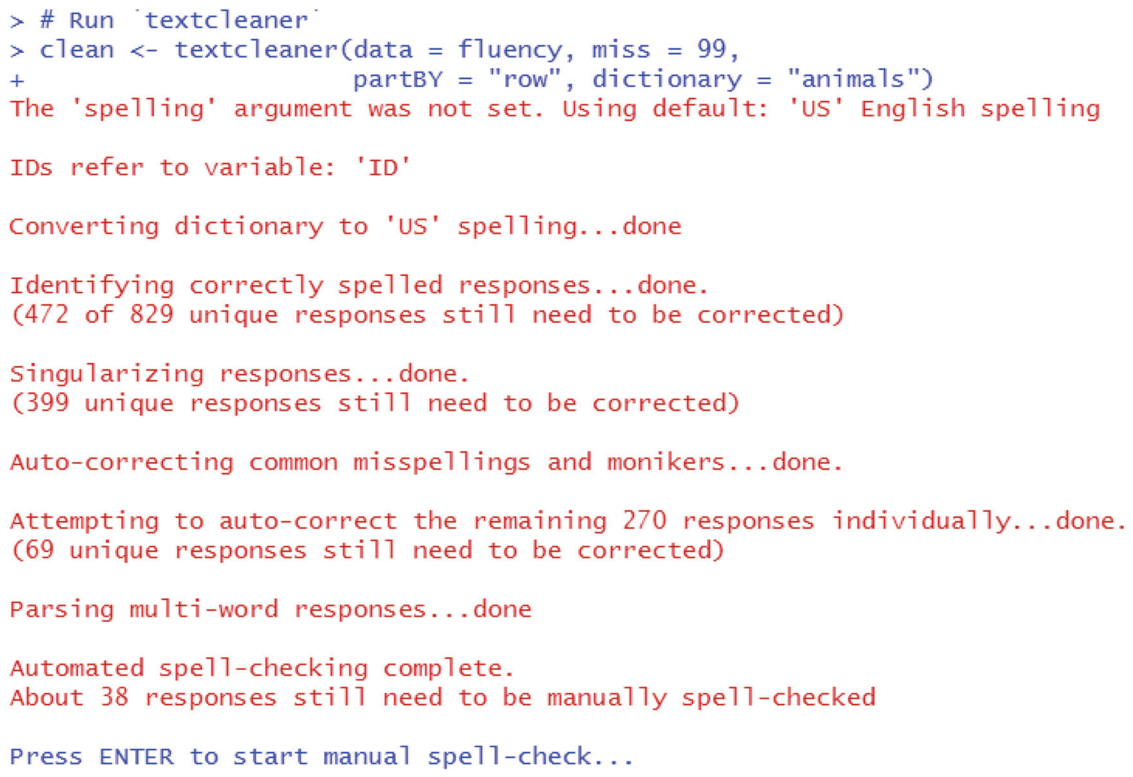
At first, the interface may be overwhelming; however, breaking down the menu will make clear that every option has its purpose and that the user is in total control over their spelling corrections. Moreover, detailed help can be provided at any time by pressing H.

Starting from the top of Figure 3, Original string refers to the raw response provided by the participant. In the example, the participant did not separate their responses by pressing ENTER when typing during the task and therefore the response appears as a string of all of their responses (i.e., multiple responses were entered as if they were a single response).

Auto-corrected string shows the participant's original response after it's been passed through the automated spell-checking process. Two things are worth noting in our example: (a) all the responses were correctly separated into individual responses and (b) *squiral* and *giraff* were auto-corrected to *squirrel* and *giraffe*, respectively.

Target word displays the word that textcleaner has prompted to be manually spell-checked. Note that this response is highlighted to draw attention to what textcleaner needs the user to manually check.

Figure 2
Automated Spell-Check and Correction Processes in Textcleaner



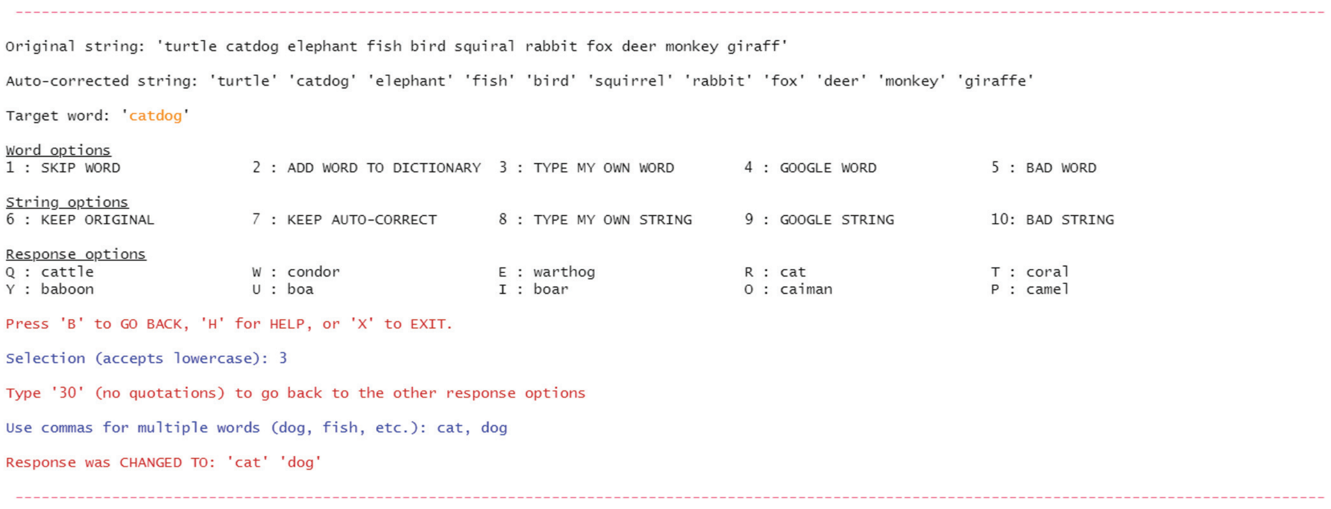
```
> # Run `textcleaner`
> clean <- textcleaner(data = fluency, miss = 99,
+                      partBY = "row", dictionary = "animals")
The 'spelling' argument was not set. Using default: 'US' English spelling

IDs refer to variable: 'ID'

Converting dictionary to 'US' spelling...done

Identifying correctly spelled responses...done.
(472 of 829 unique responses still need to be corrected)

Singularizing responses...done.
(399 unique responses still need to be corrected)

Auto-correcting common misspellings and monikers...done.

Attempting to auto-correct the remaining 270 responses individually...done.
(69 unique responses still need to be corrected)

Parsing multi-word responses...done

Automated spell-checking complete.
About 38 responses still need to be manually spell-checked

Press ENTER to start manual spell-check...
```

Note. See the online article for the color version of this figure.

Word options are options that change *only* the Target word (i.e., catdog in our example). SKIP WORD will keep the Target word “as is” and move on to the next response, ADD WORD TO DICTIONARY will keep the Target word “as is” and add it to a dictionary that can later be saved, TYPE MY OWN WORD allows the user to type their own spelling correction, GOOGLE WORD will open the user’s default Internet browser and “Google” the Target word, and BAD WORD will mark the Target word as an inappropriate response.

In contrast, String options are options that change the entire Original string. KEEP ORIGINAL and KEEP AUTO-CORRECT will change the response back the Original string or keep the

Figure 3
Example of Manual Spell-Check Interface



```
Original string: 'turtle catdog elephant fish bird squiral rabbit fox deer monkey giraff'

Auto-corrected string: 'turtle' 'catdog' 'elephant' 'fish' 'bird' 'squirrel' 'rabbit' 'fox' 'deer' 'monkey' 'giraffe'

Target word: 'catdog'

Word options
1 : SKIP WORD              2 : ADD WORD TO DICTIONARY  3 : TYPE MY OWN WORD      4 : GOOGLE WORD          5 : BAD WORD

String options
6 : KEEP ORIGINAL          7 : KEEP AUTO-CORRECT       8 : TYPE MY OWN STRING    9 : GOOGLE STRING        10: BAD STRING

Response options
Q : cattle                 W : condor                  E : warthog               R : cat                  T : coral
Y : baboon                 U : boa                     I : boar                  O : caiman               P : camel

Press 'B' to GO BACK, 'H' for HELP, or 'X' to EXIT.

Selection (accepts lowercase): 3

Type '30' (no quotations) to go back to the other response options

Use commas for multiple words (dog, fish, etc.): cat, dog

Response was CHANGED TO: 'cat' 'dog'
```

Note. See the online article for the color version of this figure.

Auto-corrected string "as is," respectively; TYPE MY OWN STRING allows the user to type response(s) that will replace the entire string (rather than just *catdog*); GOOGLE STRING will "Google" the entire string; and BAD STRING will mark all responses as inappropriate responses.

Next, Response options are 10 responses from the dictionary that were most similarly spelled (based on DL distance) to the Target word and operate as spell correction suggestions for the Target word. Below these options is a message about additional features that allow the user to "go back" to the previous response by pressing B, get detailed help for the response menu that we just described by pressing H, and save the user's progress and exit the function by pressing X.

In Figure 3, we have gone ahead and made a selection from these options where we used TYPE MY OWN WORD to correct the Target word because the participant did not hit the SPACE key to separate *cat* and *dog*. After selecting TYPE MY OWN WORD (by pressing 3) and pressing ENTER, a second prompt will appear for the user to type their own words. Multiple words can be entered when typing your own response by using a comma to separate responses. We input cat, dog and pressed ENTER. After entering a correction, a message appears to verify the response change (in our example: "cat" "dog").

Finally, it's important to reiterate that the interface depicted in Figure 3 represents the most complicated case: a response where the participant did not enter their responses separately. In most cases, the target response will not be a string of responses but rather a single response. In these cases, the interface will display Target word, Word options, and Response options only. The reader is encouraged to progress through the manual spell-check on their own. We have provided a list of our spell correction changes in our supplementary information (SI4), so that the reader can reproduce the decisions that we've made. Alternatively, the reader can load our preprocessed data by obtaining the open.preprocess data in the *SemNetCleaner* package using the following code:

```
# Load preprocessed data
clean <- open.preprocess
```

As a final step of the spell-check process, the user will be provided with instructions to verify the automated spell-check. Verification takes place in an Excel-like spreadsheet in a window that opens up after some instructions (see SI5 for our changes). At the end of the textcleaner process, a message will print letting the user know what output they can submit to our GitHub.

Textcleaner Output

With the preprocessing step complete, there are several output objects of interest for basic verbal fluency analyses and the next step of network estimation.

textcleaner's output is stored in a list object, which we designated clean in our example. These objects can be accessed using a dollar sign (e.g., clean$responses) and nested objects can be accessed within their parent object (e.g., clean$responses$original). For basic verbal fluency analyses, the object clean$behavioral contains a data frame with common behavioral measures of verbal fluency: perseverations (number of repeated responses), intrusions (number of inappropriate category exemplars), and appropriate responses (see Table 4). For network estimation, the objects clean$responses$clean and clean $responses$binary are of interest. The former is the cleaned verbal fluency responses in the order they were typed by the participant whereas the latter is a binary response matrix where participants are the rows and responses are the columns with 1's corresponding to responses a participant provided and 0's to responses they did not.

Exporting Preprocessed Data

Before exporting these matrices, it's necessary to separate our data into groups. To do so, we'll first access the grouping variable in the original dataset but also change the values to better reflect what the group values represent (i.e., 1 = low openness to experience and 2 = high openness to experience):

```
# Accessing and changing 'Group' variable
group <- ifelse(open.animals$Group == 1,
"Low", "High")
# Separate group data
low <- clean$responses$clean[which(group
== "Low"),]
high <- clean$responses$clean[which(group
== "High"),]
```

Table 4
Textcleaner Output Objects

Object	Nested object	Description
	original	Original response matrix where uppercase letters were made to lowercase and white spaces before and after responses were removed.
responses	clean	Spell-corrected response matrix where the ordering of the original responses are preserved. Inappropriate and duplicate responses have been removed.
	binary	Binary response matrix where rows are participants and columns are responses. 1's are responses given by a participant and 0's are responses not given by a participant
	correspondence	A matrix of all unique original responses provided by the participants ("from" column) and what these responses were changed to in the textcleaner process ("to" columns).
	automated	A matrix of only the unique original responses provided by participants ("from" column) that were changed ("to" columns) during the automated spell-check processes.
	manual	A matrix of only the unique original responses provided by participants ("from" column) that were changed ("to" column) during the manual spell-check process.
spellcheck	verified	A list of changes during the verification process.
behavioral	—	A data frame containing the number of perseverations, intrusions, and appropriate responses for each participant.
dictionary	—	Dictionary containing additional words added during manual spell-check. Only outputs if additional words were added.
moniker	—	Same as output as $spellcheck$manual. Duplicated for ease of saving and exporting.

To export these matrices and data frames so that they can be used in other software, the write.csv function can be used by specifying the object and path with a file name:

```
# Save cleaned responses
write.csv(low, file = "low_cleaned_res-
ponses.csv", row.names = TRUE)
write.csv(high, file = "high_cleaned_res
  ponses.csv", row.names = TRUE)
```

In the above code, we created a file called "low_cleaned.responses.csv" and "high_cleaned.responses.csv," which saved the clean response matrix to.csv files. This file can then be imported into other software to estimate and analyze semantic networks. For transparency purposes, users may also export the clean$spellcheck $manual object for colleagues and reviewers to evaluate decisions made during the manual spell-check step (see our decisions in SI4).

Finally, the convert2snafu function will save the data in a format that is compatible with the GUI for the Semantic Network and Fluency Utility (SNAFU) library (Zemla et al., 2020) in Python

```
# Save SNAFU formatted data
convert2snafu(low, high, category =
"animals")
```

This function accepts any number of groups from the same dataset and requires the category argument to be set to the verbal fluency category used. Currently, data sets must be converted one at a time so if multiple categories are preprocessed, then separate files should be created and merged to be used in SNAFU. This function will open an interactive directory explorer so the user can navigate to the desired save location as well as allow the user to input a file name. The function will handle the rest by formatting the data and saving the file as a.csv. Thus, this function allows users to seamlessly transition from this pipeline into the SNAFU GUI, facilitating cross-software compatibility for network estimation methods and analyses.

Summary

In this section we described and demonstrated how the packages *SemNetDictionaries* and *SemNetCleaner* are used to facilitate efficient and reproducible preprocessing of verbal fluency data. In this process, the raw data have been spell-checked, duplicate and inappropriate responses have been counted and removed, colloquial names for responses have been converged into one response, and cleaned response matrices have been generated. In addition, we described how to export the cleaned response matrices so that other software can be used to estimate and analyze semantic networks.

Estimating Semantic Networks

Continuing with our SemNA pipeline, the next step is to estimate semantic networks with the *SemNeT* package. The *SemNeT* package offers two options for performing this step: R console code or a point-and-click GUI using RStudio's Shiny platform. For most users, the Shiny application will be the most intuitive and straightforward to use. For more experienced R users, code to perform these analyses are provided in a script in our supplementary information (SI6). We will continue our tutorial with the Shiny application.

SemNeT Shiny Application

To open the application, the following code can be run:

```
# SemNeT Shiny Application
SemNeTShiny()
```

Running the above code will open a separate window pictured in Figure 4. If a textcleaner object and a group object (like the objects we created in the last section; clean and group, respectively) are in R's environment, then a drop-down menu and radio button to select these objects will appear (see Figure 4). Other options that will always appear are "Browse..." buttons to import a response matrix and group variable from the user's computer. If no group variable is entered, then the Shiny application will assume that the data belong to one group. Buttons to see examples of the data structures are also available. The Shiny application will give preference to data imported using the "Browse..." function. Finally, the "Load Data" button will load the data into the application. A message window will pop-up letting the user know that their data was loaded successfully.

After the data are loaded, a new tab called "Network Estimation" will open (see Figure 5), which we turn to next.

Network Estimation Methods

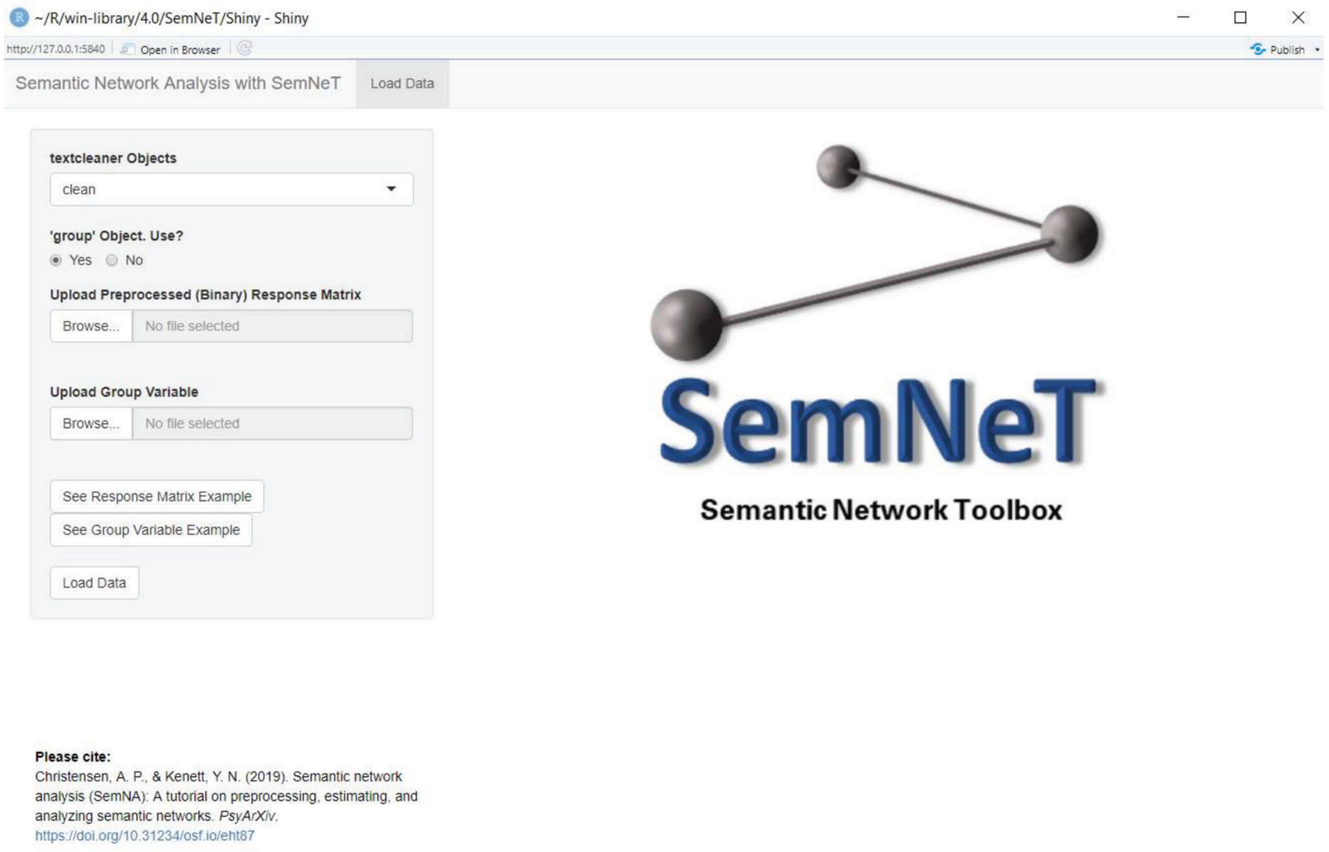
There are four network estimation methods currently available in *SemNeT*: community network (Goñi et al., 2011), naïve random walk (Lerner et al., 2009), pathfinder network (Paulsen et al., 1996; Quirin et al., 2008; Schvaneveldt, 1990), and correlation-based networks (Kenett et al., 2013). We provide brief descriptions of each of these approaches here along with the parameters that can be manipulated in the Shiny application and *SemNeT* package (for more detailed descriptions of these methods, see Zemla & Austerweil, 2018). In the Shiny GUI, tooltips are provided for each network estimation method's parameters by hovering over each parameter's input. The parameters used in the initial network estimation are used throughout the analyses in the Shiny GUI.

Community Network

The community network (CN) method estimates a semantic network using a co-occurrence matrix where two responses (e.g., *cat* and *dog*) have occurred within some *window* or distance from each response for each participant (Goñi et al., 2011). The size of the window is a parameter ("Window Size") that can be manipulated and defaults to 2 in *SemNeT*. To provide an example of this parameter, we consider a hypothetical participant's response pattern: *cat*, *fish*, *dog*, *aardvark*, *hippopotamus*, and *tardigrade*. The distance between *dog* and *cat* would be two, which would be within the window size of 2 and counted as a co-occurrence. The distance between *dog* and *tardigrade*, however, is 3 and therefore would not be counted as co-occurring.

These co-occurrence counts are then used to infer whether they are likely to occur by random chance using confidence intervals from a binomial distribution. These confidence intervals are a

Figure 4
SemNeT Shiny Application Graphical User Interface


~/R/win-library/4.0/SemNeT/Shiny - Shiny
http://127.0.0.1:5840
Open in Browser
Publish
Semantic Network Analysis with SemNeT
Load Data
Browse...
See Response Matrix Example
See Group Variable Example
Load Data
SemNeT
Semantic Network Toolbox

textcleaner Objects: clean

'group' Object. Use? [✓ Yes] No

Upload Preprocessed (Binary) Response Matrix: No file selected

Upload Group Variable: No file selected

Please cite:
Christensen, A. P., & Kenett, Y. N. (2019). Semantic network analysis (SemNA): A tutorial on preprocessing, estimating, and analyzing semantic networks. *PsyArXiv*. https://doi.org/10.31234/osf.io/eht87

Note. See the online article for the color version of this figure.

second parameter ("Significance Level") that can be adjusted using a significance level, which defaults to .05 (or a 95% confidence interval). Goñi et al. (2011) also implemented a clustering approach to "enrich" the network by connecting all nodes that are in the same cluster. This network enrichment has been implemented in *SemNeT* as a parameter ("Enrich Network") that can be changed via a drop-down menu. The drop-down menu includes FALSE and TRUE for whether the network should be enriched; the default is to not enrich the network (FALSE).

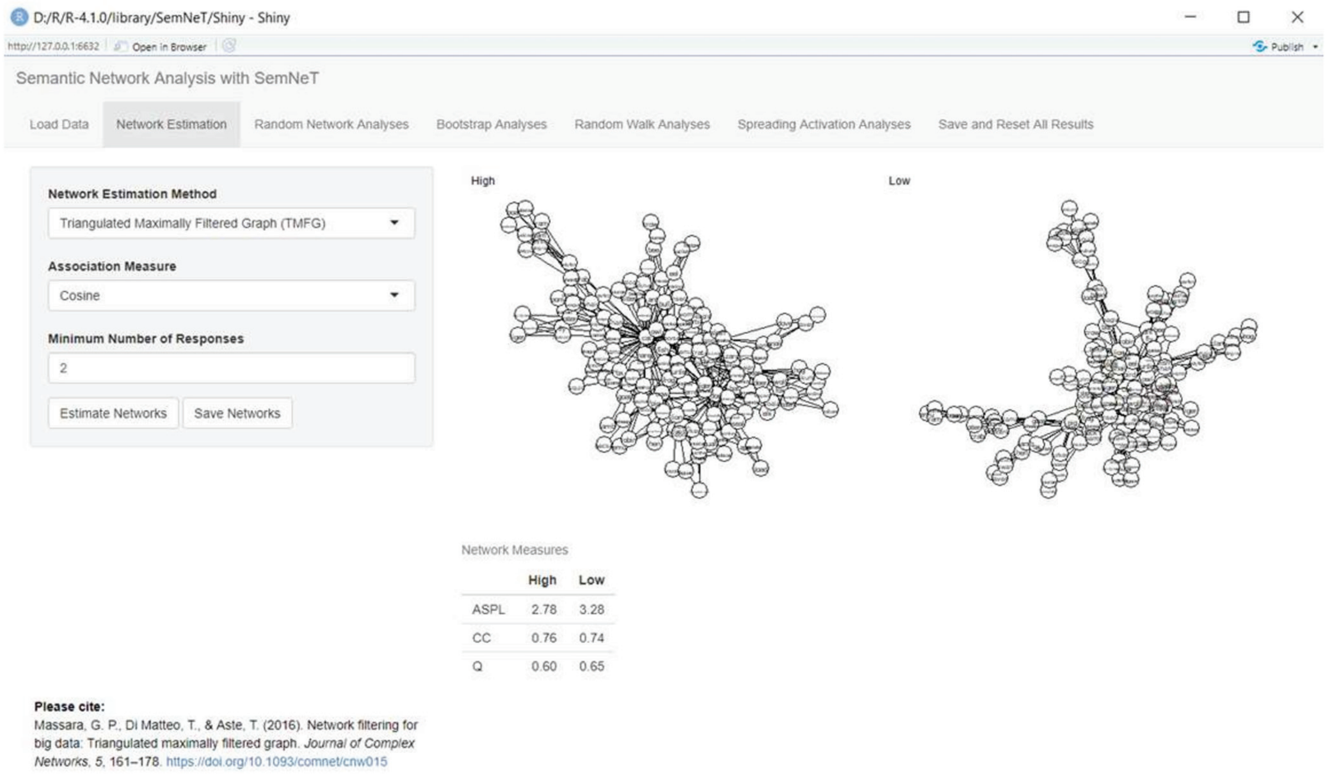
Naïve Random Walk

The naïve random walk (NRW; Lerner et al., 2009) method estimates a semantic network based on the assumption that the data are generated from an uncensored *random walk* or a stochastic process of taking "steps" from one node in the network to another. This process summarizes to estimating an edge between each adjacent pair of responses in the fluency response matrix. In this way, the NRW method operates with the assumption that adjacent responses are more likely to be related to one another. One implementation to reduce spurious co-occurrences is to count the co-occurrences of adjacent pairs across the fluency response matrix and then apply a threshold so that only a certain number or proportion of adjacent pairs that co-occur are estimated to have an edge in the network (Zemla & Austerweil, 2018). The type of threshold is the first parameter ("Type") to be selected. The two parameter options are "Number," which sets a threshold based on a certain number of co-occurrences, or "Proportion," which sets a threshold based on a certain proportion of co-occurrences in the given sample. The "Threshold" parameter can be adjusted based on the "Type" parameter selection. The default for the "Number" and "Proportion" threshold is 0. The default type of threshold for the NRW method is "Number." When applying a threshold, previous implementations have used "Number" with a threshold of 3 (Lerner et al., 2009). To our knowledge no recommendations have been made for the "Proportion" threshold.

Pathfinder Network

The pathfinder network (PN) method, as implemented in *SemNeT* estimates a semantic network using a proximity measure (e.g., Euclidean distance) between every pair of responses in the binary response matrix (Paulsen et al., 1996). The method starts by computing the proximity matrix and retains the path with the shortest distance between every pair of nodes only. There are two parameters that can be manipulated, q and r, where q constrains the number of steps between two nodes and r adjusts how the distance metric is computed. Following Zemla and Austerweil's (2018) implementation, *SemNeT* estimates the PN using the union of all *minimum spanning trees* or a set of edges that links all nodes

Figure 5
Network Comparison Plot for the Correlation-Based Network Method



Network Estimation Method: Triangulated Maximally Filtered Graph (TMFG)
Association Measure: Cosine
Minimum Number of Responses: 2

Network Measures

	High	Low
ASPL	2.78	3.28
CC	0.76	0.74
Q	0.60	0.65

Please cite:
Massara, G. P., Di Matteo, T., & Aste, T. (2016). Network filtering for big data: Triangulated maximally filtered graph. *Journal of Complex Networks, 5*, 161–178. https://doi.org/10.1093/comnet/cnw015

Note. The high openness to experience group's network is on the left and the low openness to experience group's network is on the right. ASPL = average shortest path length; CC = clustering coefficient; Q = maximum modularity coefficient. See the online article for the color version of this figure.

in the network that minimizes the total distance of all edges in the network (Quirin et al., 2008). This implementation sets the q parameter to $N - 1$ where there are no constraints on the maximum number of steps between the shortest path between two nodes and the r parameter to ∞ where the distance is the Chebyshev distance. These parameter settings of the PN method estimate the sparsest possible network. Although the q and r parameters can be manipulated, the *SemNeT* package does not implement them due to computational cost (see Quirin et al., 2008 for more details).

Correlation-Based Networks

Correlation-based network (CbN) methods constructs a semantic network based on the co-occurrence of responses across the response matrix. Using the binary response matrix, the CbN methods compute an association measure between every pair of responses, resulting in an association matrix. Most often, these association matrices are estimated using Pearson's correlation (Kenett et al., 2013); however, any association measure could be used. In the original study that used our example data, for example, cosine similarity was used to avoid negative associations between nodes in the network (for binary values, cosine similarity ranges from zero to one; Christensen, Kenett, Cotter, et al., 2018). The association measure is therefore a parameter ("Association Measure") that can be manipulated with CbN; however, Pearson's correlation and cosine similarity are recommended.

For CbN, a family of network filtering methods known as *Information Filtering Networks* (Barfuss et al., 2016), which include the minimum spanning trees that are used in the PN method (Christensen, Kenett, Aste, et al., 2018; Mantegna, 1999), are used to estimate semantic networks (Christensen, Kenett, Cotter, et al., 2018; Kenett et al., 2013). This family of filtering methods applies various geometric constraints on the association matrix to identify the most relevant information between nodes while ensuring every node is connected in the network. The two filtering methods that have been most widely applied in the literature are the *Planar Maximally Filtered Graph* (PMFG; Kenett et al., 2013; Tumminello et al., 2005) and *Triangulated Maximally Filtered Graph* (TMFG; Christensen, Kenett, Cotter, et al., 2018; Massara et al., 2016). We note that only the TMFG is implemented in the *SemNeT* package due to the time-intensiveness of the PMFG in R.[1]

Although the filters slightly differ (see Tumminello et al., 2005 and Massara et al., 2016 for PMFG and TMFG, respectively), their goals are similar: Connect nodes by maximizing the strength of their association (e.g., correlation) to other nodes while keeping the extant network *planar* or so that the network *could* be depicted on a two-dimensional surface without any edges crossing (e.g., Tumminello

[1] The PMFG method is available in R (https://github.com/AlexChristensen/PMFG), Python (SNAFU), and MATLAB (https://www.mathworks.com/matlabcentral/fileexchange/38689-pmfg?s_tid=prof_contriblnk). The R implementation is substantially slower than the Python and MATLAB implementations.

et al., 2005). This constraint retains $3n - 6$ edges in the network where n is the number of nodes in the network. The results from the PMFG and TMFG generally align with one another.

When constructing group-level networks, a common approach for CbN is to retain responses that at least two participants in each group has provided (e.g., Borodkin et al., 2016). The purpose of this constraint has been to ensure that Pearson's correlations can be computed for all response pairs and to minimize spurious associations driven by idiosyncrasies in the sample. The minimum number of responses to retain is another parameter ("Minimum Number of Responses") that can be adjusted when using the CbN method.

Finally, responses are "equated" or matched such that each group only retains the responses that are given by all other groups (Kenett et al., 2013). Equating responses ensures that all groups have the same number of nodes and therefore the same number of edges when estimating semantic networks with the CbN methods, reducing the confounding effects of different numbers of nodes and edges when making comparisons between networks (Borodkin et al., 2016; van Wijk et al., 2010).

Estimating Networks in the SemNeT Shiny Application

To continue with our tutorial, we implemented the CbN estimation method to be consistent with the analyses in the original paper (Christensen, Kenett, Cotter, et al., 2018). To be comprehensive, we performed the following analyses for each estimation method and report the results in our supplementary information (SI7). For each network estimation method, we used the *SemNeT* package's default parameter settings, which are the initialized parameter values when switching between estimation methods in the Shiny application. The reader is encouraged to estimate different networks using the "Network Estimation Method" drop-down menu. When moving forward with other analyses, the last performed network estimation method and associated parameters will be carried through all analyses. In Figure 5, we depict the comparison of each groups network for the CbN method.

When selecting a network estimation method from the drop-down menu, the parameter options will change. After a network estimation is selected and the "Estimate Networks" button is clicked, then citations for the methods will appear on the left-hand section of the screen below the parameters box. Citations will appear with each type of analysis performed in the *SemNeT* Shiny application to provide researchers with greater detail about the methods, but also to provide quick APA style references for writing the Methods section of an article (hyperlinked DOIs are provided where available).

Exporting Networks

These networks can be extracted and used in other software by pressing the "Save Networks" button, which will produce a message that allows the user to input a name for the object to be saved (e.g., "my_networks"). Upon closing out of the Shiny application, an object in R's environment called my_networks will appear. These networks will be labeled with their group name. In the example code below, we show how to extract these networks and demonstrate how to convert them to the *igraph* package's (Csardi & Nepusz, 2006) format as well as the popular network analysis and visualization software Cytoscape (Shannon et al., 2003):

```
# Extract group network results
# from Network Estimation section
low <- my_networks$network$
Lowhigh <- my_networks$network$High
# Convert network's to igraph's format
low.igraph <- convert2igraph(low)
high.igraph <- convert2igraph(high)
# Convert network's to Cytoscape's format
write.csv(
  convert2cytoscape(low),
  file = "low_network_cytoscape.csv,"
  row.names = FALSE
)
write.csv(
  convert2cytoscape(high),
  file = "high_network_cytoscape.csv,"
  row.names = FALSE
)
# Save networks for other software
write.csv(low, file = "low_network.csv",
row.names = FALSE)
write.csv(high, file = "high_network.csv",
row.names = FALSE)
```

The conversion of the networks to the *igraph* and Cytoscape formats is another demonstration of how this pipeline can be modular and facilitate cross-software compatibility. Moreover, similar to the preprocessed data export, the network objects low and high can also be saved as.csv files and exported to other software.

Summary

In this section, we briefly reviewed the network estimation methods available in the *SemNeT* package and estimated group-based semantic networks using the CbN method. For networks estimated in the *SemNeT* Shiny application, we demonstrated how they can be converted to use with the popular *igraph* package in R and Cytoscape as well as exported for other software. We continue the tutorial by applying statistical analyses to the networks estimated in this section. All statistical analyses can be completed within the *SemNeT* Shiny application and are seamlessly connected to the networks estimated in this section—that is, networks for all subsequent analyses will be estimated with the same parameters used to estimate the networks in the Network Estimation tab.

Analyzing Semantic Networks

To continue with our SemNA pipeline, the user does not need to exit the Shiny application. Instead, there are several statistical analyses available in the *SemNeT* package to apply to the networks, which appear as tabs in the Shiny application (see Figure 5). These analyses include comparisons against random networks (Random Network Analyses), bootstrapping the networks with group comparison (Bootstrap Analyses), random walks on the networks with group comparison (Random Walk Analyses), and spreading activation on individual networks provided by the *spreadr* package (Spreading Activation Analyses; Siew, 2019). For brevity, we discuss and perform a comparison against random networks and bootstrap network group comparisons (see Kenett & Austerweil, 2016 and Siew, 2019 for explanations on the Random Walk and Spreading Activation Analyses, respectively).

Global Network Measures

Global network measures are computed when the networks are estimated in the Shiny applications Network Estimation tab. Before comparing these measures between network estimation methods (see online supplementary materials), we begin with an explanation of these measures. Common SemNA metrics are global (or macroscopic) network measures. These measures focus on the structure of the entire network, emphasizing how nodes are connected as a cohesive whole (Siew, 2019). In contrast, local (or microscopic) network measures focus on the role of individual nodes in the network (Bringmann et al., 2019). Below, we briefly define and describe a few global network measures—average shortest path length, clustering coefficient, and modularity—of the many measures that are commonly used in SemNA (other global measures such as diameter and small-world index can be computed using the *igraph* or *NetworkToolbox* packages). A more extensive review of these and other network measures can be found in Siew (2019).

A common network structure that tends to emerge across many different systems (including semantic memory) is a small-world structure (Kenett et al., 2014; Lerner et al., 2009; Watts & Strogatz, 1998). Small-world networks are characterized by having a moderate average shortest path length (ASPL) and large clustering coefficient (CC). The ASPL refers to the average shortest number of steps (i.e., edges) that is needed to get between any pair of nodes in the network. That is, it's the average of the minimum number of steps from one node to all other nodes. In cognitive models, ASPL may affect the activation of associations between concepts (known as *spreading activation*; Anderson, 1983; Siew, 2019) such that a lower ASPL would increase the likelihood of reaching a greater number associations (Christensen, Kenett, Cotter, et al., 2018). Several studies of creative ability and semantic networks have linked lower ASPL to greater creative ability (Benedek et al., 2017; Kenett et al., 2014; Kenett & Faust, 2019b; but see, Lange et al., 2020). These studies have argued that the lower ASPL in the semantic network of people with higher creative ability may have allowed them to reach more remote associations, which in turn could be combined into novel and useful associations (Kenett & Faust, 2019b).

The CC refers to the extent that two neighbors of a node will be neighbors themselves, on average, in the network. A network with a higher CC, for example, suggests that nodes that are near-neighbors to each other tend to also co-occur and be connected. Wulff et al. (2018) examined younger and older adults semantic networks that were estimated using category verbal fluency data and found that the older adults had a smaller CC compared with the younger adults. This structure was interpreted as having a role in the cognitive slowing observed in older adults.

Finally, modularity measures how well a network compartmentalizes (or partitions) into subnetworks (i.e., smaller networks within the overall network; Fortunato, 2010; Newman, 2006). The maximum modularity coefficient (Q) estimates the extent to which the network has dense connections between nodes within a subnetwork and sparse (or few) connections between nodes in different subnetworks. In this implementation, Q refers to the maximum modularity possible given all possible partition organizations. These partitions are estimated using the Louvain algorithm (Blondel et al., 2008) in the *igraph* package. Higher Q values suggest that these subnetworks are more well-defined, while lower Q values suggest that the network may be less readily segmented into different parts. A few studies have demonstrated the significance of modularity in cognitive networks (Kenett et al., 2016; Siew, 2013). For example, Kenett et al. (2016) found that people with high functioning autism (Asperger syndrome) had a more modular semantic network relative to matched controls. They suggested that this "hyper" modularity might be related to rigidity of thought that often characterizes people with Asperger syndrome. In Figure 5, our results show that the high openness to experience group had a lower ASPL, higher CC, and lower Q than the low openness to experience group, suggesting they had a more flexible and interconnected semantic network.

Statistical Tests

Although global network measures provide metrics for the structure of the network, they are generally qualitative and require other approaches to statistically test for differences. Here, we implement two common procedures that are available in the *SemNeT* package that test whether these network measures are statistically different from random networks (comparisons against random networks) and between groups (bootstrap network group comparisons; Kenett et al., 2013, 2014).

Random Network Analyses

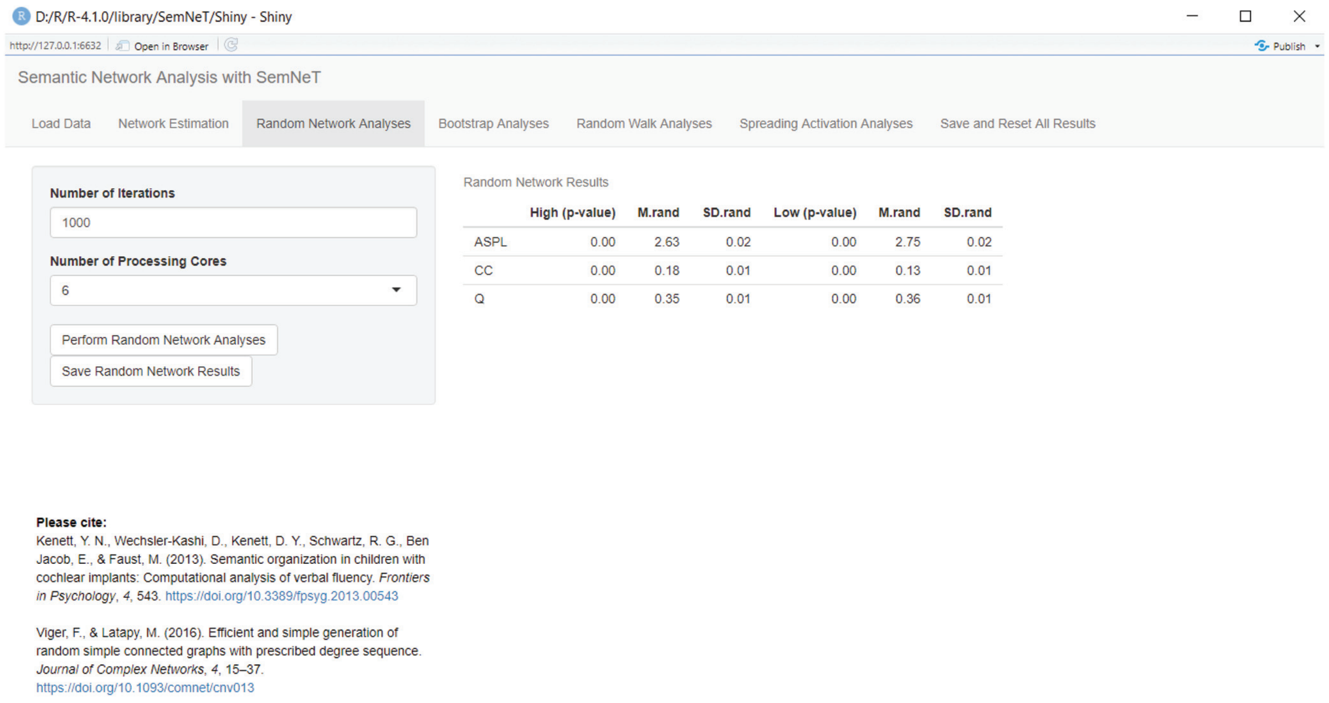
Comparisons against random networks can be performed to determine whether the network measures observed in the groups are different from what would be expected from a random network with the same number of nodes and edges (Beckage et al., 2011; Steyvers & Tenenbaum, 2005). Further, these random networks are generated such that their *degree sequence* or the number of connections to each node is preserved, maintaining the general structure of the network.

These random networks are generated using Viger and Latapy's (2016) Markov chain Monte Carlo (MCMC) algorithm on a random network with a specified degree sequence. Specifically, the approach starts with a random network with the same number of nodes and edges as the original network. The edges are than randomized using the MCMC algorithm to ensure that each node in the network preserves the number of connections it had in the original network. The random network thus serves as a null model or a test against chance that the empirical network's structure was not generated randomly. This approach is implemented using the *igraph* package's sample_degseq function.

The test against random networks approach works as follows: (a) for each group's network, generate X number of random networks (e.g., 1n000); (b) compute global network measures (i.e., ASPL, CC, and Q) for each group's random networks, resulting in a sampling distribution of these measures; and (c) compute a *p*-value for the original group's network measures based on the sampling distribution of the random network's measures (Kenett et al., 2013). Significant *p*-values ($<.05$) suggest that the empirical network's structure is different from an equivalent random network's structure. To implement this procedure, the user should click on the Random Analyses tab (see Figure 6).

There is one analysis and one computation parameter that can be changed. For the analysis parameter, the "Number of Iterations" or random networks generated can be adjusted. The default and standard in the literature has been to generate 1,000 random networks per group

Figure 6
Random Network Analyses in Shiny Application


D:/R/R-4.1.0/library/SemNeT/Shiny - Shiny
http://127.0.0.1:6632
Open in Browser
Publish
Semantic Network Analysis with SemNeT
Load Data
Network Estimation
Random Network Analyses
Bootstrap Analyses
Random Walk Analyses
Spreading Activation Analyses
Save and Reset All Results
Perform Random Network Analyses
Save Random Network Results

Number of Iterations: 1000

Number of Processing Cores: 6

Random Network Results

	High (p-value)	M.rand	SD.rand	Low (p-value)	M.rand	SD.rand
ASPL	0.00	2.63	0.02	0.00	2.75	0.02
CC	0.00	0.18	0.01	0.00	0.13	0.01
Q	0.00	0.35	0.01	0.00	0.36	0.01

Please cite:
Kenett, Y. N., Wechsler-Kashi, D., Kenett, D. Y., Schwartz, R. G., Ben Jacob, E., & Faust, M. (2013). Semantic organization in children with cochlear implants: Computational analysis of verbal fluency. *Frontiers in Psychology*, *4*, 543. https://doi.org/10.3389/fpsyg.2013.00543

Viger, F., & Latapy, M. (2016). Efficient and simple generation of random simple connected graphs with prescribed degree sequence. *Journal of Complex Networks*, *4*, 15–37. https://doi.org/10.1093/comnet/cnv013

Note. ASPL = average shortest path length; CC = clustering coefficient; Q = maximum modularity coefficient. See the online article for the color version of this figure.

network (e.g., Kenett et al., 2013). For the computation parameter, the “Number of Processors” for parallel computing can be selected. The default is for half the number of processors on the user’s machine. The more processors, the faster the computation time. After clicking the “Perform Random Network Analyses” button, a message will appear letting the user know the analysis has started and that progress can be tracked in their R console. The output from this function is a table that reports the *p*-values for each group’s network compared with the random network’s network measure values (e.g., “High (*p*-value)” and “Low (*p*-value)”) as well as the means (e.g., “M.rand”) and standard deviations (e.g., “*SD*.rand”) for the random network’s distribution of global network measures (see Figure 6). As shown in Figure 6, all global network measures were significantly different from random for both openness to experience groups. This result suggests that both networks have significantly different structures than a random network with the same number of nodes, edges, and degree sequence.

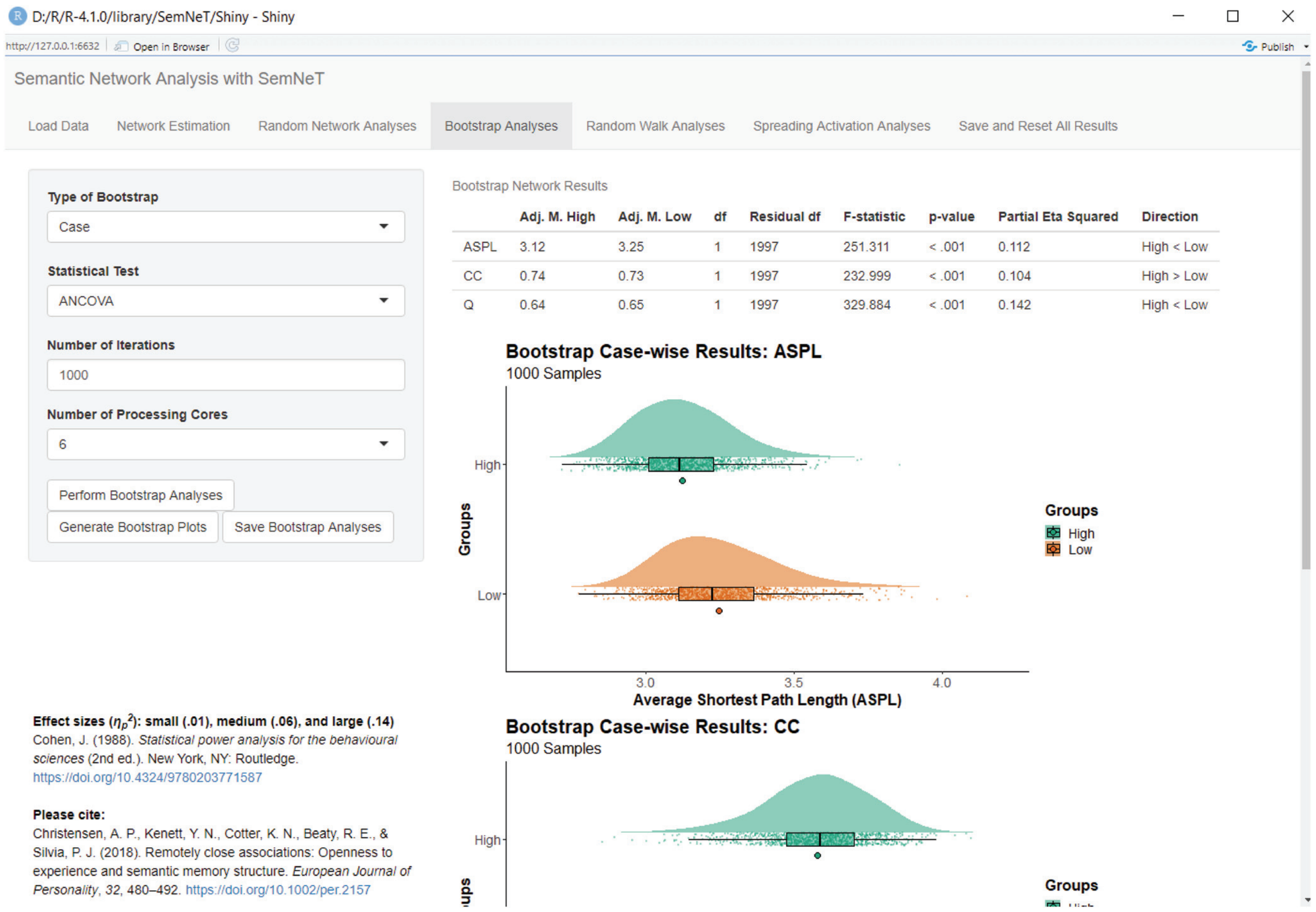
Bootstrap Analyses

The second analysis to statistically compare semantic networks applies a bootstrap method (Efron, 1979). There are two bootstrap approaches that can be applied in the SemNA pipeline: case-wise and node-wise bootstrap. The case-wise approach samples *N* participants with replacement from the respective group, meaning that some participants may be included more than once, while others may not be included at all. For each replicate sample, the network estimation method is applied and then the global network measures —ASPL, CC, and Q—are computed. This process repeats iteratively (usually 1,000 times).

The node-wise approach is available for the CbN method only because it requires that the groups’ networks consist of the exact same nodes. The node-wise approach starts by selecting a subset of nodes in the network (e.g., 50%), then estimating the networks for each group using this subset of nodes, and finally, computing the network measures for each network (Kenett et al., 2014). This method is known as *without replacement* because each node can only be selected once (Bertail, 1997; Politis & Romano, 1994; Shao, 2003). This process repeats iteratively (usually 1,000 times). The rationale for the node-wise approach is twofold: (a) if the full networks differ from each other, then any partial network consisting of the same nodes should also be different, and thus (b) the generation of many partial networks allows for a direct statistical comparison between the full networks (Kenett et al., 2016).

For both bootstrapping approaches, these bootstrap networks form sampling distributions of the global network measures, but solely based on the empirical data. These sampling distributions can then be statistically compared with a *t* test if there are only two groups being compared. If there are two or more groups, then an analysis of covariance (ANCOVA) with the number of edges used as a covariate can be used to estimating whether the global network measures are different between each group’s networks. Including edges as a covariate statistically controls for a confound that affects comparing network measures between groups. ASPL, for example, will often be smaller for networks with a greater ratio of edges to nodes (van Wijk et al., 2010). Adjusted means and effect sizes that account for this confound are then estimated. To continue with the tutorial, the user can click on the Bootstrap Analyses tab (see Figure 7).

Figure 7
Bootstrap Analyses in Shiny Application (CbN Method)


D:/R/R-4.1.0/library/SemNeT/Shiny - Shiny
http://127.0.0.1:6632
Open in Browser
Publish
Semantic Network Analysis with SemNeT
Load Data
Network Estimation
Random Network Analyses
Bootstrap Analyses
Random Walk Analyses
Spreading Activation Analyses
Save and Reset All Results
Perform Bootstrap Analyses
Generate Bootstrap Plots
Save Bootstrap Analyses
Bootstrap Case-wise Results: ASPL
1000 Samples
High
Low
Groups
3.0
3.5
4.0
Average Shortest Path Length (ASPL)
Bootstrap Case-wise Results: CC
1000 Samples
High
Groups

Type of Bootstrap: Case

Statistical Test: ANCOVA

Number of Iterations: 1000

Number of Processing Cores: 6

Bootstrap Network Results

	Adj. M. High	Adj. M. Low	df	Residual df	F-statistic	p-value	Partial Eta Squared	Direction
ASPL	3.12	3.25	1	1997	251.311	< .001	0.112	High < Low
CC	0.74	0.73	1	1997	232.999	< .001	0.104	High > Low
Q	0.64	0.65	1	1997	329.884	< .001	0.142	High < Low

Effect sizes (η_p^2): small (.01), medium (.06), and large (.14)
Cohen, J. (1988). *Statistical power analysis for the behavioural sciences* (2nd ed.). New York, NY: Routledge. https://doi.org/10.4324/9780203771587

Please cite:
Christensen, A. P., Kenett, Y. N., Cotter, K. N., Beaty, R. E., & Silvia, P. J. (2018). Remotely close associations: Openness to experience and semantic memory structure. *European Journal of Personality, 32*, 480–492. https://doi.org/10.1002/per.2157

Note. CC = clustering coefficient; Q = maximum modularity coefficient. See the online article for the color version of this figure.

Similar to the Random Network Analyses, users can select the number of bootstrap samples (i.e., “Number of Iterations”) and the “Number of Processing Cores.” Again, the default is for 1,000 replicate samples and half of the users processors, respectively. After the analyses are complete, a table with the results will appear. In Figure 7, we applied the ANCOVA approach to the CbN networks, which found the high openness to experience group had lower ASPL relative to the low openness to experience group (moderate-to-large effect size). For CC, we found the opposite: larger CC for the high openness to experience group relative to the low openness to experience group (moderate-to-large effect size). Finally, we found the high openness to experience group had lower Q relative to the low openness to experience group (large effect size). More complicated group analysis designs, such as a two-way ANOVA, can be performed using code in *SemNeT*’s test.bootSemNeT function.

In Figure 7, the CbN method’s results are displayed as an example of the output from the Shiny application. Partial eta squared effect sizes based on Cohen’s (1988) guidelines are reported with their interpretations provided in the left half of the application below the parameter selection section (see Figure 7). In addition, a button labeled “Generate Plots” will appear after the analysis is finished. Pressing this button will generate plots for the distribution of the ASPL, CC, and Q for each group. The Shiny application will offer a preview of these plots (ASPL is displayed in Figure 7) but for publication purposes larger or individual plots are often necessary. To demonstrate how to generate and access individual plots, we’ll turn to the output of the Shiny application.

Shiny Results Output

There are several options for returning output from the Shiny application. The first and recommended method is to use the Save and Reset All Results tab. In this tab, the user will find two buttons: “Save All Results” and “Clear Results.” The “Save All Results” button will prompt the user to name their output. After naming the output, the results of the last performed analyses across all analysis tabs will be saved as the user-specified named object in R’s environment after the user closes the application. The “Clear Results” button will clear all analysis results and reset the application back to the Network Estimation tab. Importantly, the verbal fluency and group data will not be cleared and any saved results prior to clearing the results will still appear in R’s environment.

The second method is to save individual results by using the “Save . . .” button within each analysis tab. Again, users will be prompted to name the object that will be saved to R’s environment. This allows users

Table 5
SemNeT Shiny Application Output Objects

Object	Description
data	Data imported into the Shiny application.
group	Group variable imported into the Shiny application.
network	The networks estimated in the Network Estimation tab. These networks will be labeled using the group variable.
measures	Network measures ASPL, CC, and Q for each group's networks estimated in the Network Estimation tab.
comparePlot	Visualization of each group's networks from the Network Estimation tab.
randomTest	Statistical results from the Random Network Analyses tab.
bootstrap	The bootstrapped samples and measures from the Bootstrap Analyses tab.
bootstrapTest	Statistical results table from the Bootstrap Analyses tab.
bootstrapPlot	Plots of the statistical results from the Bootstrap Analyses tab.
randomWalk	Results from the Random Walk Analyses tab.
spreadingActivation	Results from the Spreading Activation Analyses tab.
spreadingActivationPlot	Plots of the results in the Spreading Activation Analyses tab.

Note. ASPL = average shortest path length; CC = clustering coefficient; Q = maximum modularity coefficient.

to perform and save the results of different network estimation methods or analyses without switching or clearing the results of the application.

Finally, as a fail-safe, all data and results from the last performed analyses in the application will be saved in an object called resultShiny in R's environment upon closing the application (i.e., clicking on the "X" in the top right corner of the window). A message will print letting the user know that this has been done. Below is a table (see Table 5) describing the possible output:

The objects included in the output will depend what analyses were performed and how the results were saved. By default, data, group, network, measures, and comparePlot will always be output with the other analyses. If, for example, we estimated all analyses and went into the Bootstrap Network Analyses tab and clicked the "Save Bootstrap Analyses" button, then a list containing only the default and bootstrap, bootstrapTest, and bootstrapPlot objects will be saved (other analysis objects will not be included). If the "Save All Results" button in the Save and Reset All Results tab is used, then all data and analyses will be saved.

By exiting the application, specific analysis objects can be accessed, such as the bootstrap plots that were generated in our last example, using the standard $structure mentioned previously. The following code can be entered into R's console to access and plot individual bootstrap plots:

```
# Plot bootstrap results
## ASPL
plot(resultShiny$bootstrapPlot$aspl)
## CC
plot(resultShiny$bootstrapPlot$cc)
## Q
plot(resultShiny$bootstrapPlot$q)
```

Each network measure is output as an individual plot, which allows the user to manipulate each plot into the desired size for presentation and publication (e.g., making sure plot features are large enough to be easily discernible). Moreover, each plot is generated using the *ggplot2* package (Wickham, 2016), which means that they can be manipulated to use different colors (e.g., distributions, boxplots, dots) and adjust titles, axis labels, and legend keys (see http://www.cookbook-r.com/Graphs/ for getting started).

Summary

In this section, we briefly reviewed two statistical analyses techniques, comparisons against random networks and bootstrap network group comparisons, available in the *SemNeT* package. We then demonstrated how to perform these analyses using the Shiny application and provided an overview of how the output can be saved. From this output, we demonstrated how to obtain individual plots from the bootstrap results that enables the user to generate publication ready figures.

SemNA and SNAFU Comparison

We introduce a comprehensive SemNA pipeline that can be applied sequential or modular fashion. The modular nature of the pipeline opens the doors for compatibility with other software. One closely related software is the SNAFU or the Semantic Network and Fluency Utility (Zemla et al., 2020) library in Python. SNAFU was developed specifically for verbal fluency semantic network analysis and provides many of the same behavioral and network measures available in SemNA (see Table 6). Similarly, both SemNA and SNAFU are capable of investigating group- and individual-based networks. There are three key differences between the software: (a) SemNA's preprocessing capabilities are more thorough and can be applied to tasks other than verbal fluency (e.g., free association and semantic similarity); (b) SemNA contains statistical analyses for comparing networks; and (c) SemNA is designed to be compatible with many other network analysis software including SNAFU. Although not demonstrated in our article, SemNA's statistical analyses can handle more complicated designs including two-way ANOVA and longitudinal designs. In sum, SemNA is designed to be a flexible and comprehensive tool for semantic network analysis whereas SNAFU is designed to be specific to verbal fluency semantic network analysis.

Discussion

In this article, we put forward a SemNA pipeline for preprocessing, estimating, and analyzing semantic networks. This pipeline was accompanied by three R packages—*SemNetDictionaries*, *SemNetCleaner*, and *SemNeT*—that are designed together to facilitate the application of semantic network analysis. With these packages, we demonstrated how this pipeline could be applied to real-world data by reanalyzing verbal fluency data previously collected by Christensen, Kenett, Cotter, et al. (2018). To get to these results, this tutorial went step-by-step through preprocessing the verbal fluency data by checking for spelling errors and inappropriate and duplicate responses, and

Table 6
Comparison of SemNA and SNAFU

Features	SemNA	SNAFU
Software	R	Python
Focus	Group-based networks	Individual-based networks
Preprocessing	Yes (extensive)	Yes
Behavioral analyses	Intrusions, perseverations, total, and unique responses	Intrusions, preservations, total responses, clustering, switching
Network estimation methods	Pathfinder, correlation-based, naïve random walk, community	Pathfinder, correlation-based, naïve random walk, community, U-INVITE, First Edge
Network measures	Average shortest path length, clustering coefficient, modularity, and others (see compatibility)	Average shortest path length, clustering coefficient, small-worldness, degree distribution, density
Statistical analyses	Bootstrap, random network	—
Spreading activation analyses	Random walk, spreading activation (spreadr)	—
GUI	Yes	Yes
Compatibility (software)	igraph (R), qgraph (R), NetworkToolbox (R), spreadr (R), SNAFU (Python), Cytoscape	—

Note. SemNA = semantic network analysis; SNAFU = Semantic Network and Fluency Utility; GUI = graphical user interface.

then loading the data into the *SemNeT* Shiny application to estimate and analyze semantic networks.

This tutorial serves two roles not yet filled in the literature: First, for novice SemNA researchers, we provided a step-by-step resource for how to execute SemNA with verbal fluency data; second, for experienced SemNA researchers, we provided a standardized approach to efficiently preprocess verbal fluency data, while also increasing the transparency of this process. Because the pipeline is modular, novice and experienced SemNA researchers alike are encouraged to interchange parts of the pipeline with other software, maximizing the potential analyses available for their verbal fluency data.

The standardization of the preprocessing stage offers a few advances for semantic network analysis and beyond. First, the preprocessing of semantic data becomes more consistent and transparent across researchers, encouraging open science practices (e.g., output can be shared in the peer-review process; see SI4 and SI5). Such an approach enables common practices across labs and fields to emerge. Moreover, with the rise of big data and crowd-sourcing (Mandera et al., in press), norms can be developed for each verbal fluency category (e.g., De Deyne et al., 2019), which would further facilitate the automation and accuracy of preprocessing verbal fluency data. The *SemNetDictionaries* package, for example, provides researchers with the capability to create their own dictionaries, which can stimulate the development of these norms. Broadly, researchers are encouraged to share their dictionaries and monikers on our GitHub, so that future versions can include them as predefined glossaries in the package.

Second, the *SemNetCleaner* package enables researchers to efficiently preprocess their verbal fluency data into a format that can be used with any semantic network estimation method available in *SemNeT* and SNAFU (Zemla & Austerweil, 2018; Zemla et al., 2020). The speed with which *SemNetCleaner* preprocesses verbal fluency data is magnitudes times faster than any manual method. Take, for instance, our example data: there were over 9,000 responses and over 800 unique responses. Although many of these responses were spelled correctly, the user was required to make fewer than 50 decisions. With *SemNetCleaner*, the entire preprocessing step can be managed in under 20 min.

Beyond SemNA, the automated preprocessing step can be used to achieve more reliable verbal fluency results because human error is largely removed from this process. This means that researchers who are not necessarily interested in performing SemNA can still use *SemNetCleaner* to ensure the reliability and validity of their verbal fluency results (e.g., total number of appropriate responses). This becomes especially important when using automated methods of scoring for verbal fluency (e.g., clustering and switching, semantic and lexical diversity; Kim et al., 2019; Pakhomov et al., 2016; Zemla et al., 2020).

Finally, the *SemNeT* package offers researchers a few methods for the analysis of semantic network data (including some methods not covered in this article such as random walk analysis; Kenett & Austerweil, 2016). One notable analysis comes from the *spreadr* package (Siew, 2019), which simulates spreading activation across a network. Spreading activation is an important concept in the semantic memory literature, which may provide insights into how information is retrieved from semantic memory. This pipeline can seamlessly be integrated with *spreadr* by using the *SemNeT* Shiny application for network estimation and statistical analyses (i.e., Spreading Activation Analyses tab). With the networks outputted from the network estimation stage, researchers can also feasibly implement other network analysis packages such as *igraph*, *qgraph* (Epskamp et al., 2012), and *NetworkToolbox* (Christensen, 2018) in R to compute other network measures not covered in this tutorial (e.g., centrality measures).

Importantly, while we focused our analyses on two groups, SemNA is fully capable of analyzing more complicated such as two-way ANOVA, longitudinal, and within-person designs. These more complicated designs are less feasibly accomplished in the *SemNeT* Shiny application and should be executing using R code. Nonetheless, the preprocessing stage can be carried out on all designs with the data being separated into the designs structure afterward. In sum, this suite of open-source R packages provides a broad set of flexible tools for researchers to conduct SemNA efficiently and feasibly while producing reliable and reproducible SemNA results.

Best Practices and Open Questions

More generally, several open questions related to the semantic network analysis best practices remain currently debated. To our knowledge, there has only been one simulation study that evaluated different network estimation methods (Zemla & Austerweil, 2018).[2]

[2] For a review and discussion of this simulation, please see the supplementary information (SI8)

There is a pressing need to identify best practices for each network estimation method and their parameters. The network estimation functions in *SemNeT* and the GUI use their default values that have been used by their seminal articles, yet it remains unclear whether these are the optimal parameters. Simulation studies are necessary to determine how these methods perform under different conditions and whether some should be preferred over others.

Key questions for simulations to answer are: What sample sizes are necessary for semantic network analysis? Some studies have used as few as 30 people per group (Rastelli et al., 2020), while others have used more than 200 people per group (Christensen, Kenett, Cotter, et al., 2018). To our knowledge, there are no quantitative procedures available that can determine power for network analysis. What is the effect of different data generating mechanisms for participant levels (individuals vs. groups) and tasks (fluency, free association, and semantic similarity; Hills & Kenett, 2021)? Do different network estimation parameters perform better for certain tasks or conditions? How should discrepancies between methods be interpreted? These questions underlie deeper theoretical questions (Hills & Kenett, 2021): Do individuals have different network structures and different processes? How are group-level networks related to individual-level networks? How should edges in semantic networks be defined? Are semantic networks stable or vary over time? While a thorough discussion of these issues is outside the scope of this tutorial article, the reader is referred to Hills and Kenett (2021) for a current discussion of these issues.

Some avenues forward, besides simulation studies, might be to estimate all four methods and compare them as we've done. The pattern of our results demonstrates some consensus between estimation methods (see online supplementary materials), suggesting that consensus results could be used for the most reliable interpretations. Another avenue is to preregister one network estimation method that has been used in related work. This approach at least keeps consistency between investigations and can allow for between-sample inferences. Some work, for example, has used group-level correlation-based networks of fluency data for the semantic networks of highly creative people, finding they have more interconnected (higher ASPL) and flexible (lower Q) structures (Kenett et al., 2014; Kenett et al., 2016; Li et al., 2021; Rastelli et al., 2020). Although our software contributes to standardizing the semantic network pipeline, we urge researchers to work toward developing best practices to guide new and expert researchers on what methods to use.

SemNA Future Directions

There are several future directions specific to the SemNA pipeline. The first is to expand the functionality of the statistical and network analyses to include free association and semantic similarity tasks. In present form, the preprocessing packages can be used to provide some automaticity of free association and semantic similarity tasks. Future versions of the SemNA pipeline should accommodate a larger number of analyses that can be applied to free association and semantic similarity tasks. Expanding dictionaries to include other categories but also more languages will further increase the accessibility of the software to researchers globally. Using a community-driven database hosted on GitHub, the SemNA pipeline can incorporate researcher-derived dictionaries into SemNA to facilitate the integration of other categories and languages. Finally, it should accommodate more complex designs (e.g., longitudinal, within-subject), and additional types of data (e.g., free associations, semantic relatedness), in a more user-friendly and compatible way.

Conclusion

With this pipeline in hand, researchers can readily apply SemNA to their own data, opening doors to applications of SemNA in new domains. The strength of analyzing verbal fluency data is that it is quick and easy to administer within any experimental paradigm. For applied researchers, this makes verbal fluency data an attractive option for studying cognitive processes associated with behavioral and psychopathological phenomena. Indeed, some promising avenues for future research are to understand how the structure and processes of semantic memory are associated with psychopathological disorders (Elvevåg et al., 2017; Holmlund et al., 2019; Kenett & Faust, 2019a; Kenett et al., 2016) and personality traits (Christensen, Kenett, Cotter, et al., 2018). To date, there have been only a handful of studies that have quantitatively examined the semantic memory processes that underlie these and other psychological phenomena. By quantitatively analyzing semantic networks, cognitive theory can be extended and integrated into psychological theories, bridging these sometimes disparate yet parallel lines of research.

References

Anderson, J. R. (1983). A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, *22*(3), 261–295. https://doi.org/10.1016/S0022-5371(83)90201-3

Ardila, A., Ostrosky-Solís, F., & Bernal, B. (2006). Cognitive testing toward the future: The example of semantic verbal fluency (ANIMALS). *International Journal of Psychology*, *41*(5), 324–332. https://doi.org/10.1080/00207590500345542

Barfuss, W., Massara, G. P., Di Matteo, T., & Aste, T. (2016). Parsimonious modeling with information filtering networks. *Physical Review E*, *94*(6-1), 062306. https://doi.org/10.1103/PhysRevE.94.062306

Baronchelli, A., Ferrer-i-Cancho, R., Pastor-Satorras, R., Chater, N., & Christiansen, M. H. (2013). Networks in cognitive science. *Trends in Cognitive Sciences*, *17*(7), 348–360. https://doi.org/10.1016/j.tics.2013.04.010

Beckage, N., Smith, L., & Hills, T. (2011). Small worlds and semantic network growth in typical and late talkers. *PLoS ONE*, *6*(5), e19348. https://doi.org/10.1371/journal.pone.0019348

Benedek, M., Kenett, Y. N., Umdasch, K., Anaki, D., Faust, M., & Neubauer, A. C. (2017). How semantic memory structure and intelligence contribute to creative thought: A network science approach. *Thinking & Reasoning*, *23*(2), 158–183. https://doi.org/10.1080/13546783.2016.1278034

Bertail, P. (1997). Second-order properties of an extrapolated bootstrap without replacement under weak assumptions. *Bernoulli*, *3*(2), 149–179. https://doi.org/10.2307/3318585

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10), P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008

Borge-Holthoefer, J., & Arenas, A. (2010). Semantic networks: Structure and dynamics. *Entropy*, *12*(5), 1264–1302. https://doi.org/10.3390/e12051264

Borodkin, K., Kenett, Y. N., Faust, M., & Mashal, N. (2016). When pumpkin is closer to onion than to squash: The structure of the second language lexicon. *Cognition*, *156*(2016), 60–70. https://doi.org/10.1016/j.cognition.2016.07.014

Borsboom, D., & Cramer, A. O. J. (2013). Network analysis: An integrative approach to the structure of psychopathology. *Annual Review of*

Clinical Psychology, *9*, 91–121. https://doi.org/10.1146/annurev-clinpsy-050212-185608

Botvinik-Nezer, R., Holzmeister, F., Camerer, C. F., Dreber, A., Huber, J., Johannesson, M., & Schonberg, T. (2020). Variability in the analysis of a single neuroimaging dataset by many teams. *Nature*, *582*(7810), 84–88. https://doi.org/10.1038/s41586-020-2314-9

Bousfield, W. A., & Sedgewick, C. H. W. (1944). An analysis of sequences of restricted associative responses. *The Journal of General Psychology*, *30*(2), 149–165. https://doi.org/10.1080/00221309.1944.10544467

Bringmann, L. F., Elmer, T., Epskamp, S., Krause, R. W., Schoch, D., Wichers, M., Wigman, J. T. W., & Snippe, E. (2019). What do centrality measures measure in psychology networks? *Journal of Abnormal Psychology*, *128*(8), 892–903. https://doi.org/10.1037/abn0000446

Christensen, A. P. (2018). NetworkToolbox: Methods and measures for brain, cognitive, and psychometric network analysis in R. *The R Journal*, *10*(2), 422–439. https://doi.org/10.32614/RJ-2018-065

Christensen, A. P., Kenett, Y. N., Aste, T., Silvia, P. J., & Kwapil, T. R. (2018). Network structure of the Wisconsin Schizotypy Scales–Short Forms: Examining psychometric network filtering approaches. *Behavior Research Methods*, *50*(6), 2531–2550. https://doi.org/10.3758/s13428-018-1032-9

Christensen, A. P., Kenett, Y. N., Cotter, K. N., Beaty, R. E., & Silvia, P. J. (2018). Remotely close associations: Openness to experience and semantic memory structure. *European Journal of Personality*, *32*(4), 480–492. https://doi.org/10.1002/per.2157

Cohen, J. (1988). *Statistical power analysis for the behavioural sciences* (2nd ed.). Routledge. https://doi.org/10.4324/9780203771587

Collins, A. M., & Loftus, E. F. (1975). A spreading-activation theory of semantic processing. *Psychological Review*, *82*(6), 407–428. https://doi.org/10.1037/0033-295X.82.6.407

Cramer, A. O. J., Van Der Sluis, S., Noordhof, A., Wichers, M., Geschwind, N., Aggen, S. H., Kendler, K. S., & Borsboom, D. (2012). Dimensions of normal personality as networks in search of equilibrium: You can't like parties if you don't like people. *European Journal of Personality*, *26*(4), 414–431. https://doi.org/10.1002/per.1866

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal, Complex Systems*, *1695*, 1–9. https://www.semanticscholar.org/paper/The-igraph-software-package-for-complex-network-Cs/\%C3/\%A1rdi-Nepusz/1d2744b83519657f5f2610698a8ddd177ced4f5c?p2df

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7*(3), 171–176. https://doi.org/10.1145/363958.363994

De Deyne, S., Kenett, Y. N., Anaki, D., Faust, M., & Navarro, D. J. (2016). Large-scale network representations of semantics in the mental lexicon. In M. N. Jones (Ed.), *Big data in cognitive science: From methods to insights* (pp. 174–202). Taylor & Francis. https://doi.org/10.4324/9781315413570

De Deyne, S., Navarro, D. J., Perfors, A., Brysbaert, M., & Storms, G. (2019). The "Small World of Words" English word association norms for over 12,000 cue words. *Behavior Research Methods*, *51*(3), 987–1006. https://doi.org/10.3758/s13428-018-1115-7

DeYoung, C. G., Quilty, L. C., & Peterson, J. B. (2007). Between facets and domains: 10 aspects of the Big Five. *Journal of Personality and Social Psychology*, *93*(5), 880–896. https://doi.org/10.1037/0022-3514.93.5.880

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, *7*(1), 26. https://doi.org/10.1214/aos/1176344552

Elvevåg, B., Foltz, P. W., Rosenstein, M., Ferrer-i-Cancho, R., De Deyne, S., Mizraji, E., & Cohen, A. (2017). Thoughts about disordered thinking: Measuring and quantifying the laws of order and disorder. *Schizophrenia Bulletin*, *43*(3), 509–513. https://doi.org/10.1093/schbul/sbx040

Epskamp, S., Cramer, A. O. J., Waldorp, L. J., Schmittmann, V. D., & Borsboom, D. (2012). qgraph: Network visualizations of relationships in psychometric data. *Journal of Statistical Software*, *48*(4), 1–18. https://doi.org/10.18637/jss.v048.i04

Feinerer, I., Hornik, K., & Meyer, D. (2008). Text mining infrastructure in R. *Journal of Statistical Software*, *25*(5), 1–54. https://doi.org/10.18637/jss.v025.i05

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, *486*(3-5), 75–174. https://doi.org/10.1016/j.physrep.2009.11.002

Fried, E. I., & Cramer, A. O. J. (2017). Moving forward: Challenges and directions for psychopathological network theory and methodology. *Perspectives on Psychological Science*, *12*(6), 999–1020. https://doi.org/10.1177/1745691617705892

Goñi, J., Arrondo, G., Sepulcre, J., Martincorena, I., Vélez de Mendizábal, N., Corominas-Murtra, B., Bejarano, B., Ardanza-Trevijano, S., Peraita, H., Wall, D. P., & Villoslada, P. (2011). The semantic organization of the animal category: Evidence from semantic verbal fluency and network theory. *Cognitive Processing*, *12*(2), 183–196. https://doi.org/10.1007/s10339-010-0372-x

Hills, T. T., & Kenett, Y. N. (2021). Is the mind a network? Maps, vehicles, and skyhooks in cognitive network science. *Topics in Cognitive Science*. Advance online publication. https://doi.org/10.1111/tops.12570

Holmlund, T. B., Cheng, J., Foltz, P. W., Cohen, A. S., & Elvevåg, B. (2019). Updating verbal fluency analysis for the 21st century: Applications for psychiatry. *Psychiatry Research*, *273*, 767–769. https://doi.org/10.1016/j.psychres.2019.02.014

Jun, K.-S., Zhu, X., Rogers, T., Yang, Z., & Yuan, M. (2015). Human memory search as initial-visit emitting random walk. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 1072–1080). NeurIPS Proceedings. https://proceedings.neurips.cc/paper/2015/hash/dc6a70712a252123c40d2adba6a11d84-Abstract.html

Karuza, E. A., Thompson-Schill, S. L., & Bassett, D. S. (2016). Local patterns to global architectures: Influences of network topology on human learning. *Trends in Cognitive Sciences*, *20*(8), 629–640. https://doi.org/10.1016/j.tics.2016.06.003

Kenett, Y. N., Anaki, D., & Faust, M. (2014). Investigating the structure of semantic networks in low and high creative persons. *Frontiers in Human Neuroscience*, *8*, 407. https://doi.org/10.3389/fnhum.2014.00407

Kenett, Y. N., & Austerweil, J. L. (2016). *Examining search processes in low and high creative individuals with random walks.* Proceedings of the 38th annual meeting of the cognitive science society, Austin, TX. https://cogsci.mindmodeling.org/2016/papers/0066/index.html

Kenett, Y. N., Beaty, R. E., Silvia, P. J., Anaki, D., & Faust, M. (2016). Structure and flexibility: Investigating the relation between the structure of the mental lexicon, fluid intelligence, and creative achievement. *Psychology of Aesthetics, Creativity, and the Arts*, *10*(4), 377–388. https://doi.org/10.1037/aca0000056

Kenett, Y. N., & Faust, M. (2019a). Clinical cognitive networks: A graph theory approach. In M. S. Vitevitch (Ed.), *Network science in cognitive science* (pp. 136–165). Routledge.

Kenett, Y. N., & Faust, M. (2019b). A semantic network cartography of the creative mind. *Trends in Cognitive Sciences*, *23*(4), 271–274. https://doi.org/10.1016/j.tics.2019.01.007

Kenett, Y. N., Gold, R., & Faust, M. (2016). The hyper-modular associative mind: A computational analysis of associative responses of persons with Asperger syndrome. *Language and Speech*, *59*(Pt. 3), 297–317. https://doi.org/10.1177/0023830915589397

Kenett, Y. N., Wechsler-Kashi, D., Kenett, D. Y., Schwartz, R. G., Ben Jacob, E., & Faust, M. (2013). Semantic organization in children with cochlear implants: Computational analysis of verbal fluency. *Frontiers in Psychology*, *4*, 543. https://doi.org/10.3389/fpsyg.2013.00543

Kim, N., Kim, J.-H., Wolters, M. K., MacPherson, S. E., & Park, J. C. (2019). Automatic scoring of semantic fluency. *Frontiers in Psychology*, *10*, 1020. https://doi.org/10.3389/fpsyg.2019.01020

Kumar, A. A., Steyvers, M., & Balota, D. A. (2021). A critical review of network-based and distributional approaches to semantic memory structure and processes. *Topics in Cognitive Science*. Advance online publication. https://doi.org/10.1111/tops.12548

Lange, K. V., Hopman, E. W., Zemla, J. C., & Austerweil, J. L. (2020). Evidence against a relation between bilingualism and creativity. *PLoS ONE*, *15*(6), e0234928. https://doi.org/10.1371/journal.pone.0234928

Lerner, A. J., Ogrocki, P. K., & Thomas, P. J. (2009). Network graph analysis of category fluency testing. *Cognitive and Behavioral Neurology*, *22*(1), 45–52. https://doi.org/10.1097/WNN.0b013e318192ccaf

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, *10*, 707–710.

Li, Y., Kenett, Y. N., Hu, W., & Beaty, R. E. (2021). Flexible semantic network structure supports the production of creative metaphor. *Creativity Research Journal*, *33*(3), 209–223. https://doi.org/10.1080/10400419.2021.1879508

Mandera, P., Keuleers, E., & Brysbaert, M. (in press). Recognition times for 62 thousand English words: Data from the English Crowdsourcing Project. *Behavior Research Methods*. https://doi.org/10.31234/osf.io/wm5gv

Mantegna, R. N. (1999). Hierarchical structure in financial markets. *The European Physical Journal B Condensed Matter and Complex Systems*, *11*(1), 193–197. https://doi.org/10.1007/s100510050929

Massara, G. P., Di Matteo, T., & Aste, T. (2016). Network filtering for big data: Triangulated maximally filtered graph. *Journal of Complex Networks*, *5*, 161–178. https://doi.org/10.1093/comnet/cnw015

McCrae, R. R., & Costa, P. T. (2007). Brief versions of the NEO PI-3. *Journal of Individual Differences*, *28*(3), 116–128. https://doi.org/10.1027/1614-0001.28.3.116

Nelson, D. L., McEvoy, C. L., & Schreiber, T. A. (2004). The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, *36*(3), 402–407. https://doi.org/10.3758/BF03195588

Newman, M. E. J. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, *103*(23), 8577–8582. https://doi.org/10.1073/pnas.0601602103

Ooms, J. (2018). *hunspell: High-performance stemmer, tokenizer, and spell checker*. https://CRAN.R-project.org/package=hunspell

Pakhomov, S. V. S., Eberly, L., & Knopman, D. (2016). Characterizing cognitive performance in a large longitudinal study of aging with computerized semantic indices of verbal fluency. *Neuropsychologia*, *89*, 42–56. https://doi.org/10.1016/j.neuropsychologia.2016.05.031

Paulsen, J. S., Romero, R., Chan, A., Davis, A. V., Heaton, R. K., & Jeste, D. V. (1996). Impairment of the semantic network in schizophrenia. *Psychiatry Research*, *63*(2-3), 109–121. https://doi.org/10.1016/0165-1781(96)02901-0

Politis, D. N., & Romano, J. P. (1994). Large sample confidence regions based on subsamples under minimal assumptions. *The Annals of Statistics*, *22*(4), 2031–2050. https://doi.org/10.1214/aos/1176325770

Quirin, A., Cordón, O., Guerrero-Bote, V. P., Vargas-Quesada, B., & Moya-Anegón, F. (2008). A quick MST-based algorithm to obtain pathfinder networks (∞, n - 1). *Journal of the American Society for Information Science and Technology*, *59*(12), 1912–1924. https://doi.org/10.1002/asi.20904

Rastelli, C., Greco, A., & Finocchiaro, C. (2020). Revealing the role of divergent thinking and fluid intelligence in children's semantic memory organization. *Journal of Intelligence*, *8*(4), 43. https://doi.org/10.3390/jintelligence8040043

Rinker, T. W. (2020). *qdap: Quantitative discourse analysis package*. https://github.com/trinker/qdap

Schvaneveldt, R. W. (1990). *Pathfinder associative networks: Studies in knowledge organization*. Ablex Publishing.

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., & Ideker, T. (2003). Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research*, *13*(11), 2498–2504. https://doi.org/10.1101/gr.1239303

Shao, J. (2003). Impact of the bootstrap on sample surveys. *Statistical Science*, *18*(2), 191–198. https://doi.org/10.1214/ss/1063994974

Siew, C. S. Q. (2013). Community structure in the phonological network. *Frontiers in Psychology*, *4*, 553. https://doi.org/10.3389/fpsyg.2013.00553

Siew, C. S. Q. (2019). spreadr: An R package to simulate spreading activation in a network. *Behavior Research Methods*, *51*(2), 910–929. https://doi.org/10.3758/s13428-018-1186-5

Siew, C. S. Q., Wulff, D. U., Beckage, N. M., Kenett, Y. N., & Meštrović, A. (2019). Cognitive network science: A review of research on cognition through the lens of network representations, processes, and dynamics. *Complexity*, *2019*, 1–24. https://doi.org/10.1155/2019/2108423

Silberzahn, R., Uhlmann, E. L., Martin, D. P., Anselmi, P., Aust, F., Awtrey, E., Bahník, Š., Bai, F., Bannard, C., Bonnier, E., Carlsson, R., Cheung, F., Christensen, G., Clay, R., Craig, M. A., Dalla Rosa, A., Dam, L., Evans, M. H., Flores Cervantes, I., . . . Nosek, B. A. (2018). Many analysts, one data set: Making transparent how variations in analytic choices affect results. *Advances in Methods and Practices in Psychological Science*, *1*(3), 337–356. https://doi.org/10.1177/2515245917747646

Stella, M., Beckage, N. M., Brede, M., & De Domenico, M. (2018). Multiplex model of mental lexicon reveals explosive learning in humans. *Scientific Reports*, *8*(1), 2259. https://doi.org/10.1038/s41598-018-20730-5

Steyvers, M., & Tenenbaum, J. B. (2005). The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive Science*, *29*(1), 41–78. https://doi.org/10.1207/s15516709cog2901_3

Tumminello, M., Aste, T., Di Matteo, T., & Mantegna, R. N. (2005). A tool for filtering information in complex systems. *Proceedings of the National Academy of Sciences of the United States of America*, *102*(30), 10421–10426. https://doi.org/10.1073/pnas.0500298102

Unsworth, N., Spillers, G. J., & Brewer, G. A. (2011). Variation in verbal fluency: A latent variable analysis of clustering, switching, and overall performance. *Quarterly Journal of Experimental Psychology*, *64*(3), 447–466. https://doi.org/10.1080/17470218.2010.505292

van Wijk, B. C. M., Stam, C. J., & Daffertshofer, A. (2010). Comparing brain networks of different size and connectivity density using graph theory. *PLoS ONE*, *5*(10), e13701. https://doi.org/10.1371/journal.pone.0013701

Viger, F., & Latapy, M. (2016). Efficient and simple generation of random simple connected graphs with prescribed degree sequence. *Journal of Complex Networks*, *4*(1), 15–37. https://doi.org/10.1093/comnet/cnv013

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*(6684), 440–442. https://doi.org/10.1038/30918

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer. https://ggplot2-book.org/

Wulff, D. U., De Deyne, S., Jones, M. N., Mata, R., & Aging Lexicon Consortium. (2019). New perspectives on the aging lexicon. *Trends in Cognitive Sciences*, *23*(8), 686–698. https://doi.org/10.1016/j.tics.2019.05.003

Wulff, D. U., Hills, T., & Mata, R. (2018). Structural differences in the semantic networks of younger and older adults. *PsyArXiv*. https://doi.org/10.31234/osf.io/s73dp

Zemla, J. C., & Austerweil, J. L. (2018). Estimating semantic networks of groups and individuals from fluency data. *Computational Brain & Behavior*, *1*(1), 36–58. https://doi.org/10.1007/s42113-018-0003-7

Zemla, J. C., Cao, K., Mueller, K. D., & Austerweil, J. L. (2020). SNAFU: The semantic network and fluency utility. *Behavior Research Methods*, *52*(4), 1681–1699. https://doi.org/10.3758/s13428-019-01343-w

Received December 6, 2020
Revision received August 17, 2021
Accepted October 18, 2021 ■