Supplementary Information

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

SI 1. R Session Information

```
R version 4.1.0 (2021-05-18)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 19043)

Matrix products: default

locale:
[1] LC_COLLATE=English_United States.1252
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] shinyBS_0.61             shinyMatrix_0.6.0        shinyalert_2.0.0
 [4] shinyjs_2.0.0            shiny_1.6.0              SemNeT_1.4.3
 [7] SemNetCleaner_1.3.3      SemNetDictionaries_0.1.9 kableExtra_1.3.4
[10] knitr_1.33               knitcitations_1.0.12     papaja_0.1.0.9997
```

```
loaded via a namespace (and not attached):
 [1] tidyselect_1.1.1  xfun_0.25         purrr_0.3.4       colorspace_2.0-2
 [5] generics_0.1.0    vctrs_0.3.8       htmltools_0.5.1.1 viridisLite_0.4.0
 [9] yaml_2.2.1        utf8_1.2.2        rlang_0.4.11      later_1.2.0
[13] pillar_1.6.2      glue_1.4.2        DBI_1.1.1         lifecycle_1.0.0
[17] plyr_1.8.6        stringr_1.4.0     munsell_0.5.0     gtable_0.3.0
[21] rvest_1.0.1       evaluate_0.14     fastmap_1.1.0     httpuv_1.6.1
[25] fansi_0.5.0       Rcpp_1.0.7        xtable_1.8-4      promises_1.2.0.1
[29] scales_1.1.1      webshot_0.5.2     jsonlite_1.7.2    mime_0.11
[33] systemfonts_1.0.2 ggplot2_3.3.5     png_0.1-7         digest_0.6.27
[37] stringi_1.7.3     bookdown_0.23     dplyr_1.0.7       grid_4.1.0
[41] tools_4.1.0       magrittr_2.0.1    tibble_3.1.3      RefManageR_1.3.0
[45] crayon_1.4.1      pkgconfig_2.0.3   ellipsis_0.3.2    xml2_1.3.2
[49] lubridate_1.7.10  assertthat_0.2.1  rmarkdown_2.10    svglite_2.0.0
[53] httr_1.4.2        rstudioapi_0.13   R6_2.5.0          compiler_4.1.0
```

SI 2. SemNA Manuscript R Script

```
############################################################
#### Semantic Networks Estimated from Verbal Fluency ####
############################################################

# Install packages (and their dependencies)
install.packages(c("SemNetDictionaries", "SemNetCleaner", "SemNeT"),
                 dependencies = c("Imports", "Suggests"))

# Shiny packages
install.packages(c("shiny", "shinyjs", "shinyalert",
                   "shinyMatrix", "shinyBS"))

# Load packages
library(SemNetDictionaries)
library(SemNetCleaner)
library(SemNeT)

################################################
#### 1. Preprocessing Verbal Fluency Data ####
################################################

#####################################
## 1.1. SemNetDictionaries Package ##
#####################################
```

```
#--------------------------------#
# 1.1.1. Pre-defined dictionaries #
#--------------------------------#

# Check for available dictionaries
dictionaries()

# Load 'animals' dictionary
load.dictionaries("animals")

# Load all words starting with 'f'
load.dictionaries("f")

# Load multiple dictionaries
load.dictionaries("fruits", "vegetables")

#----------------------------#
# 1.1.2. Custom dictionaries #
#----------------------------#

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

?append.dictionary
```

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

################################
## 1.2. SemNetCleaner Package ##
################################

# Data from 'SemNetCleaner'
data("open.animals")

## Fluency data
fluency <- open.animals[,-c(1:2)]

# Run `textcleaner`
clean <- textcleaner(data = fluency, miss = 99,
                     partBY = "row", dictionary = "animals")

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

```r
# Get groups
group <- ifelse(open.animals$Group == 1, "Low", "High")



#######################################################
#### Estimating and Analyzing Semantic Networks ####
#######################################################


# Run SemNeT Shiny application
SemNeTShiny()
```

SI 3. SemNA Template R Script

```
############################################################
#### Semantic Networks Estimated from Verbal Fluency ####
############################################################

# Install packages (and their dependencies)
install.packages(c("SemNetDictionaries", "SemNetCleaner", "SemNeT"),
                 dependencies = c("Imports", "Suggests"))

# Shiny packages
install.packages(c("shiny", "shinyjs", "shinyalert",
                   "shinyMatrix", "shinyBS"))

# Load packages
library(SemNetDictionaries)
library(SemNetCleaner)
library(SemNeT)

#-----------------#
# Loading in data #
#-----------------#

# Your own data
my.data <- read.data()

# `read.data` will automatically load in data
```

```
# from common file extensions such as
# Excel, R, Matlab, and SPSS
#
# An interactive menu will allow you to navigate
# to the file you'd like to load and will
# load the data in the correct format for
# the pipeline

#-----------------#
# Data management #
#-----------------#

# If your data has variables that are not
# subject IDs and verbal fluency responses,
# then you should remove these variables before
# running the preprocessing step
#
# To figure out which variables you need to remove
# you can use the `colnames` function to identify
# these columns
colnames(my.data)

# Identify the numbers of the columns that need to
# be removed
#
# For example:
prep.data <- my.data[,-c(1:4, 67, 89)]
```

```
# The function `c` is concatenate, which
# creates a vector with the numbers typed
# into between the parentheses.
#
# The `[1,1]` represents elements with
# the rows on the left of the comma and
# columns on the right of the comma. In the example,
# `[1,1]`, this would be element in row 1 and column 1.
# If there is no number input for either row or column
# (e.g., `[,1]`), then the entire row or column is selected
# (e.g., all of column `1` is selected).
#
# The `:` means all numbers in-between. So, in the
# above example, we've selected columns 1 *through*
# 4, 67, and 89.
#
# The `-` removes rows or columns from the matrix.
#
# The resulting matrix should consist ONLY of
# your subject IDS and raw verbal fluency responses.
#
# If you have a grouping variable in this data,
# it is OKAY to remove it. We will come back to this later.


##############################################
#### 1. Preprocessing Verbal Fluency Data ####
```

```
##############################################


################################
## 1.2. SemNetCleaner Package ##
################################


#-------------------#
# 1.2.1. Spell check #
#-------------------#


# Documentation for `textcleaner` function
?textcleaner


# Run 'textcleaner'
clean <- textcleaner(data = prep.data, miss = 99,
                     partBY = "row", dictionary = "animals")


#-----------------#
# `data` argument #
#-----------------#
##
## This is where you can input `prep.data`


#-----------------#
# `miss` argument #
#-----------------#
##
```

```
## This argument is for missing data.
## If you have missing data and use a different
## value that NA, NaN, or empty cells, then
## an additional value can be assigned.
##
## Note that NA, NaN, and empty cells are
## automatically identified as missing data


#-------------------#
# `partBY` argument #
#-------------------#
##
## If your data has participants' responses going
## across the rows (from left to right),
## then this argument should be "row"
##
## If your data has participants' responses going
## down the columns (from top to bottom),
## then this argument should be "col"


#----------------------#
# `dictionary` argument #
#----------------------#
##
## This argument specifies the dictionaries
## to be used for the automated cleaning
## and manual spelling suggestion for
```

```
## the data.
##
## To see which dictionaries are available,
## type and enter: dictionaries()
##
## For letter fluency tasks,
## single letters can be used
## For example: dictionary = c("f")
##
## If no dictionary represents the appropriate
## category for spell-check, then
## the general dictionary will be used.


# Get directory to save in
dir <- choose.dir()


# Save .csv of manual changes
write.csv(clean$spellcheck$manual,
          paste(dir, "manual_changes.csv", sep = "/"),
          row.names = TRUE)



# Verbal fluency response totals
totals <- clean$behavioral$Appropriate


#######################################################
#### Estimating and Analyzing Semantic Networks ####
```

```
#####################################################


# Run SemNeT Shiny application
SemNeTShiny()
```

SI 4. Manual Chages Made in `textcleaner`

	Option Selected	Target Response	Changed To
1	3	catdog	cat, dog
2	5	moze	NA
3	8	did	bear
4	5	creatures	NA
5	3	catefrog	cat, frog
6	5	criters	NA
7	5	mario	NA
8	Q	chiuaua	chihuahua
9	W	garafi	giraffe
10	5	snack	NA
11	5	jesus	NA
12	Q	squrill	squirrel
13	5	your mom	NA
14	Q	geaniu pig	guinea pig
15	Q	crocidle	crocodile
16	Q	dinasor	dinosaur
17	Q	buffel	buffalo
18	Q	doplin	dolphin
19	5	more cats	NA
20	5	lotus	NA
21	Q	ostrig	ostrich
22	Q	pingwin	penguin
23	Q	amardillo	armadillo
24	Q	merekat sp	meerkat
25	Q	heghog	hedgehog

26	5	lagoon	NA
27	Q	getcho sp	gecko
28	5	oh my	NA
29	5	bablefish	NA
30	W	analope	antelope
31	5	sporian	NA
32	I	women	human
33	Q	lizers	lizard
34	W	koal	koala
35	W	atalope	antelope
36	W	teranchilla	tarantula
37	5	manster	NA
38	Q	sprikbok	springbok

SI 5. Changes made during `textcleaner` verification

```
$house

          from    to

Previous  "house" "mouse"

Corrected "house" "NA"


$beasts

          from     to

Previous  "beasts" "yeast"

Corrected "beasts" "NA"


$money

          from    to

Previous  "money" "monkey"

Corrected "money" "NA"


$god

          from  to

Previous  "god" "cod"

Corrected "god" "NA"


$chia

          from   to

Previous  "chia" "cheetah"

Corrected "chia" "NA"


$`sugar bear`
```

```
          from         to
Previous  "sugar bear" "sugar glider"
Corrected "sugar bear" "NA"


$at
          from to
Previous  "at" "gnat"
Corrected "at" "NA"


$cantelope
          from        to
Previous  "cantelope" "antelope"
Corrected "cantelope" "NA"


$lamp
          from   to
Previous  "lamp" "lamb"
Corrected "lamp" "NA"
```

SI 6. R Console Code for Different Network Estimation Methods

```
############################################################
#### Semantic Networks Estimated from Verbal Fluency ####
############################################################

# Install packages (and their dependencies)
install.packages(c("SemNetDictionaries", "SemNetCleaner", "SemNeT"),
                 dependencies = c("Imports", "Suggests"))

# Load packages
library(SemNetDictionaries)
library(SemNetCleaner)
library(SemNeT)

################################################
#### 1. Preprocessing Verbal Fluency Data ####
################################################

#####################################
## 1.1. SemNetDictionaries Package ##
#####################################

#---------------------------------#
# 1.1.1. Pre-defined dictionaries #
#---------------------------------#
```

```
# Check for available dictionaries
dictionaries()


# Load 'animals' dictionary
load.dictionaries("animals")


# Load all words starting with 'f'
load.dictionaries("f")


# Load multiple dictionaries
load.dictionaries("fruits", "vegetables")


#----------------------------#
# 1.1.2. Custom dictionaries #
#----------------------------#


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


?append.dictionary
```

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

################################
## 1.2. SemNetCleaner Package ##
################################

# Data from 'SemNetCleaner'
data("open.animals")

## Fluency data
fluency <- open.animals[,-c(1:2)]

# Run `textcleaner`
clean <- textcleaner(data = fluency, miss = 99,
                     partBY = "row", dictionary = "animals")

# Load preprocessed data
clean <- open.preprocess

# Get groups
group <- ifelse(open.animals$Group == 1, "Low", "High")

########################################
```

```
#### 2. Estimating Semantic Networks ####
##########################################


##################
## 2.1. Process ##
##################


#----------------------------------------#
# 2.1.1. Preparation for network estimation #
#----------------------------------------#


# Attach 'Group' variable to the binary response matrix
behav <- cbind(open.animals$Group, clean$responses$binary)


# For Community and Pathfinder networks:
## behav <- cbind(open.animals$Group, clean$responses$clean)


# Create low and high openness to experience response matrices
low <- behav[which(behav[,1]==1),-1]
high <- behav[which(behav[,1]==2),-1]


# Save binary response matrices
write.csv(low, "low_BRM.csv", row.names = TRUE)
write.csv(high, "high_BRM.csv", row.names = TRUE)


#--------------------------#
# 2.1.2. Network estimation #
```

```
#---------------------------#

## Community Network
net.low <- CN(low)
net.high <- CN(high)

# For other networks:
##
## Naive Random Walk
### net.low <- NRW(low)
### net.high <- NRW(high)
##
## Pathfinder Network
### net.low <- PF(low)
### net.high <- PF(high)
##
## TMFG
### Finalize matrices so that each response
### has been given by at least two participants
### final.low <- finalize(low, minCase = 2)
### final.high <- finalize(high, minCase = 2)
###
### Equate the responses across the networks
### eq <- equate(final.low, final.high)
### equate.low <- eq$final.low
### equate.high <- eq$final.high
### Compute cosine similarity for the 'low' and
```

```
### 'high' equated binary response matrices
### cosine.low <- similarity(equate.low, method = "cosine")
### cosine.high <- similarity(equate.high, method = "cosine")
###
### Estimate 'low' and 'high' openness to experience networks
### net.low <- TMFG(cosine.low)
### net.high <- TMFG(cosine.high)

# Save the networks
write.csv(net.low, "low_network.csv", row.names = FALSE)
write.csv(net.high, "high_network.csv", row.names = FALSE)

########################################
#### 3. Analyzing Semantic Networks ####
########################################

##############################################
## 3.1. Visualization of Semantic Networks ##
##############################################

# Visually compare networks
compare_nets(net.low, net.high,
             title = list("Low Openness", "High Openness"),
             config = "spring", weighted = FALSE)

#################################
## 3.2. Global Network Measures ##
```

```
##################################

# Compute network measures
semnetmeas(net.low, meas = c("ASPL", "CC", "Q"), weighted = FALSE)
semnetmeas(net.high, meas = c("ASPL", "CC", "Q"), weighted = FALSE)

############################
## 3.3. Statistical Tests ##
############################

#-------------------------------------#
# 3.3.1. Tests against random networks #
#-------------------------------------#

# Compute tests against random networks
rand.test <- randnet.test(net.low, net.high, iter = 1000, cores = 4)

#--------------------------------------#
# 3.3.2. Bootstrapped case-wise networks #
#--------------------------------------#

# Compute bootstrap network analysis
boot <- bootSemNeT(low, high, method = "CN",
                   type = "case", iter = 1000,
                   cores = 4)

# See documentation: ?bootSemNeT
```

```
# For other methods:
## Naive Random Walk
### boot <- bootSemNeT(low, high, method = "NRW",
###                    type = "case", iter = 1000,
###                    cores = 4)
##
## Pathfinder Network
### boot <- bootSemNeT(low, high, method = "PF",
###                    type = "case", iter = 1000,
###                    cores = 4)
##
## TMFG
### boot <- bootSemNeT(low, high, method = "TMFG",
###                    type = "case", iter = 1000,
###                    sim = "cosine", cores = 4)

# Perform t-tests on bootstrap results
tests <- test.bootSemNeT(boot)

# Plot bootstrap results
plots <- plot(boot, groups = c("Low","High"),
              measures = c("ASPL", "CC", "Q"))

# To arrange plots so they are stacked
install.packages("gridExtra")
library(gridExtra)
```

```
grid.arrange(plots$aspl, plots$cc, plots$q)


#--------------------------------------#
# 3.3.2. Bootstrapped node-wise networks #
#--------------------------------------#


# Compute partial bootstrap network analysis
## 50% of nodes remaining in network
boot.fifty <- bootSemNeT(low, high, method = "TMFG", type = "node",
                         percent = .50, iter = 1000,
                         sim = "cosine", cores = 4)
## 60% of nodes remaining in network
boot.sixty <- bootSemNeT(low, high, method = "TMFG", type = "node",
                         percent = .60, iter = 1000,
                         sim = "cosine", cores = 4)
## 70% of nodes remaining in network
boot.seventy <- bootSemNeT(low, high, method = "TMFG", type = "node",
                           percent = .70, iter = 1000,
                           sim = "cosine", cores = 4)
## 80% of nodes remaining in network
boot.eighty <- bootSemNeT(low, high, method = "TMFG", type = "node",
                          percent = .80, iter = 1000,
                          sim = "cosine", cores = 4)
## 90% of nodes remaining in network
boot.ninety <- bootSemNeT(low, high, method = "TMFG", type = "node",
                          percent = .90, iter = 1000,
                          sim = "cosine", cores = 4)
```

```
# Perform t-tests on bootstrap results
tests <- test.bootSemNeT(boot.fifty, boot.sixty, boot.seventy,
                         boot.eighty, boot.ninety)

# Plot bootstrap results
plots <- plot(boot.fifty, boot.sixty, boot.seventy,
              boot.eighty, boot.ninety, groups = c("Low","High"),
              measures = c("ASPL", "CC", "Q"))

# To arrange plots so they are stacked
install.packages("gridExtra")
library(gridExtra)
grid.arrange(plots$aspl, plots$cc, plots$q)
```

SI 7. Comparison of All Network Estimation Methods

Across the network estimation methods, ASPL was lower for the high openness to experience group relative to the low openness to experience group except for the NRW method (Table 7). Similarly, CC was lower for the high openness to experience group relative to the low openness to experience group except for the CbN method. Finally, Q was lower for the high openness to experience group for the PN and CbN methods, higher for the NRW method, and equivalent for the CN method relative to the low openness to experience group.

Table 7

Global Networks Measures for Each Network Estimation Method

		Network Estimation Method			
Measure	**Group**	CN	NRW	PN	CbN
ASPL	High	5.17	2.74	5.09	2.78
	Low	5.35	2.66	5.32	3.28
CC	High	0.20	0.31	0.53	0.76
	Low	0.26	0.32	0.54	0.74
Q	High	0.74	0.22	0.22	0.60
	Low	0.74	0.20	0.25	0.65

In our example, there were two groups, so the result was based on random networks computed for each group's network structure. As shown in Table 8, all global network measures were significantly different from random for both openness to experience groups across all network estimation methods. This result suggests that both networks have significantly different structures than a random network with the same number of nodes, edges, and degree sequence.

Table 8

Random Network Analysis for Each Network Estimation Method

		Group			
		High		Low	
Method	**Measure**	*M*	*SD*	*M*	*SD*
CN	ASPL	4.22	0.07	4.10	0.09
	CC	0.02	0.01	0.03	0.01
	Q	0.57	0.01	0.57	0.01
NRW	ASPL	2.52	0.01	2.44	0.02
	CC	0.34	0.01	0.38	0.02
	Q	0.18	0.00	0.17	0.00
PN	ASPL	2.04	0.00	2.-5	0.00
	CC	0.68	0.01	0.65	0.01
	Q	0.06	0.00	0.07	0.00
CbN	ASPL	2.62	0.02	2.75	0.02
	CC	0.18	0.01	0.13	0.01
	Q	0.35	0.01	0.36	0.01

Note. All means (*M*) and standard deviations (*SD*) are for the distributions of the random networks. All *p*-values were $< .001$

We performed the case-wise bootstrap for all network estimation measures and report the results below.

Table 9

ANCOVA for the Case-wise Bootstrap Analysis of Each Network Estimation Method

		Adjusted M			
Method	**Measure**	High	Low	$F_{1,1997}$	η_p^2
CN	ASPL	3.28	3.31	61.32	0.03
	CC	0.27	0.28	24.99	0.01
	Q	0.49	0.51	597.61	0.23
NRW	ASPL	2.78	2.70	1082.65	0.35
	CC	0.28	0.29	431.35	0.18
	Q	0.23	0.22	1726.97	0.46
PN	ASPL	4.93	5.17	230.39	0.10
	CC	0.52	0.50	738.42	0.27
	Q	0.17	0.19	595.54	0.23
CbN	ASPL	3.12	3.24	245.54	0.11
	CC	0.74	0.73	254.12	0.11
	Q	0.64	0.65	326.41	0.14

Note. All p-values were $< .001$. η_p^2 effect sizes following Cohen (1988): small = 0.01, moderate = 0.06, and large = 0.14

In Table 9, we applied the ANCOVA approach for all network estimation methods, which found the high openness to experience group had lower ASPL relative to the low openness to experience group (small to moderately large effect sizes) except for the NRW method (large effect size). Two methods, CN and NRW, found the high openness to experience group had lower CC relative to the low openness to experience group (small and large effect sizes, respectively). Conversely, the PN and CbN methods found the opposite (larger CC for the high openness to experience group relative to the low openness to experience group; large and moderately large effect sizes, respectively). Finally, all network estimation methods found the high openness to experience group had lower Q relative to the low openness to experience group (all large effect sizes) except for the NRW method (large effect size).

SI 8. Review and Discussion of Zemla and Austerweil (2018)

To our knowledge, there is only one study to systematically compare these and other network estimation methods for group- and individual-level semantic networks estimated from category verbal fluency data (Zemla & Austerweil, 2018). In Zemla and Austerweil's (2018) study, they obtained animal category exemplars from the University of South Florida free association norms and generated category verbal fluency networks using censored random walks (Nelson, McEvoy, & Schreiber, 2004). In addition, they obtained human-rated semantic similarity for animals pairs and evaluated the average similarity of these ratings for edges that were present and absent in the networks estimated with these methods.

Based on their results, they recommended the CN method for groups where the network may not be fully connected and the goal is to minimize non-edge (absent edge) similarities; the PN method was recommended for groups where the network is fully connected and the goal is to maximize edge similarities whereas the NRW method was recommended when the goal is to minimize non-edge similarities. Finally, two methods that are not available in *SemNeT* but are available in SNAFU (Zemla, Cao, Mueller, & Austerweil, 2020), U-INVITE and Hierarchical U-INVITE, were recommended for groups when the network is fully connected and the goal is to minimize non-edge similarities and when the network may not be fully connected and the goal is to maximize edge similarity, respectively. CbN method were not recommended for any of these conditions because they were the only method to have non-edge similarities that were greater than their edge similarities.

This systematic comparison is an important first step for evaluating these methods but more studies are needed. Comparisons of network estimation methods are beyond the scope of this paper, so we provide users flexibility in the estimation method they prefer. However, we provide two recommendations for future comparisons. First, the type of verbal fluency data (e.g., free association, phonological, category) should be generated based on the same data type. Although the USF network is a well-validated network that "accurately

reflects mental associations between items in the network" (Zemla & Austerweil, 2018, p. 47), it would be surprising if the cognitive processes (e.g., strategic search, spreading activation) of free association norms were identical to the processes of generating category exemplars. Some processes are likely to be similar, but there are different task constraints and people are likely using different strategies in their search processes (e.g., Unsworth, Spillers, & Brewer, 2011). Therefore, methods should be assessed for each type of verbal fluency task using data generated from the respective task.

Second, the category verbal fluency data were generated using a censored random walk, which may be a reasonable data generating mechanism (Jun, Zhu, Rogers, Yang, & Yuan, 2015), but it matches the way in which the U-INVITE method estimates networks. This potentially confounds the results to the extent in which the data generating process matches the way some methods estimate networks—that is, it's plausible that the U-INVITE and methods based on random walks (e.g., NRW) performed well because the data generating model was highly similar to how the networks are estimated. Therefore, different data generating mechanisms should be used to test the generality of these methods and how they are affected should the true data generating model not correspond to a censored random walk. In sum, more systematic investigations into how these network estimations perform under different conditions (e.g., task type, data generating mechanism) are necessary to posit any methodological recommendations beyond Zemla and Austerweil's (2018) findings.