Epistemic Network Analysis and Ordered Network Analysis in Learning Analytics

Yuanru Tan, Zachari Swiecki, A. R. Ruis, and David Shaffer

1 Introduction

This chapter provides a tutorial on conducting *epistemic network analysis* (ENA) and *ordered network analysis* (ONA) using R. We introduce these two techniques together because they share similar theoretical foundations, but each addresses a different challenge for analyzing large-scale qualitative data on learning processes.

ENA and ONA are methods for quantifying, visualizing, and interpreting network data. Taking coded data as input, ENA and ONA represent associations between codes in undirected or directed weighted network models, respectively. Both techniques measure the strength of association among codes and illustrate the structure of connections in network graphs, and they quantify changes in the composition and strength of those connections over time. Importantly, ENA and ONA enable comparison of networks both visually and via summary statistics, so they can be used to explore a wide range of research questions in contexts where patterns of association in coded data are hypothesized to be meaningful and where comparing those patterns across individuals or groups is important.

In the following sections, we will (1) briefly review literature relevant to the application of ENA and ONA, (2) provide a step-by-step guide to implementing ENA and ONA in R, and (3) suggest additional resources and examples for further exploration. By the end of this chapter, readers will be able to apply these techniques in their own research.

Y. Tan (✉) · A. R. Ruis · D. Shaffer
University of Wisconsin, Madison, WI, USA
e-mail: yuanru.tan@wisc.edu

Z. Swiecki
Monash University, Clayton, Australia

M. Saqr, S. López-Pernas (eds.), *Learning Analytics Methods and Tutorials*,
https://doi.org/10.1007/978-3-031-54464-4_18

2 Literature Review

2.1 *Epistemic Network Analysis (ENA)*

ENA is a method for identifying and quantifying connections in coded data and representing them in undirected weighted network models [1]. There are two key features that differentiate ENA from other networks analysis tools or multivariate analyses: (1) ENA produces summary statistics that can be used to compare the differences in the content of networks rather than just their structure; and (2) ENA network visualizations provide information that is mathematically consistent with those summary statistics, which facilitates meaningful interpretation of statistical differences [2]. These features enable researchers to analyze a wide range of phenomena in learning analytics, including complex thinking and knowledge construction [3, 4], collaborative problem solving [5, 6], socio-emotional aspects of learning [7], mentoring [8], and teacher professional development [9–11].

One key feature that makes ENA an effective method in modeling collaborative interaction is that ENA can model individuals' unique contributions to collaborative discourse *while accounting for* group context, and thus both individuals *and* groups can be analyzed in the same model. This feature is particularly valuable in collaborative learning environments, where the interactions and contributions of each individual are related and should not be treated as a series of isolated events. For example, Swiecki et al. [6] analyzed the communications of air defense warfare teams in training exercises and found that ENA was not only able to reveal differences in individual performance identified in a qualitative analysis of the collaborative discourse, but also to test those differences statistically.

2.2 *Ordered Network Analysis (ONA)*

Ordered Network Analysis (ONA) extends the theoretical and analytical advantages of ENA to account for the order of events by producing *directed* weighted networks rather than undirected models [12]. Like ENA, ONA takes coded data as input, identifies and measures connections among coded items, and visualizes the structure of connections in a metric space that enables both statistical and visual comparison of networks. However, ONA models the order in which codes appear in the data, enabling analysis of phenomena in which the order of events is hypothesized to be important.

For example, Tan et al. [12] used ONA to model the performance of military teams learning to identify, assess, and respond to potential threats detected by radar. The findings demonstrate that ONA could detect qualitative differences between teams in different training conditions that were not detected with unordered models and show that they are statistically significant. In their work, Tan et al. [12] argued that ONA possesses an advantage over methods such as Sequential Pattern Mining

(SPM), which is widely used to identify frequent sequential patterns. In contrast to SPM, which prioritizes the specific micro-sequential order of events, ONA models processes by accounting for the co-temporal order of interactions between the units of analysis in response and what they are responding to. Consequently, ONA is a more appropriate methodological choice when modeling processes in ill-formed problem-solving scenarios, where collaborative interactions do not follow a prescribed sequence of steps but where the order of activities is still important.

ONA has also been used to analyze log data from online courses. For example, Fan et al. [13] analyzed self-regulated learning tactics employed by learners in Massive Open Online Courses (MOOC) using ONA and process mining. The authors found that ONA provided more nuanced interpretations of learning tactics compared to process mining because ONA models learning tactics across four dimensions: frequency, continuity, order, and the role of specific learning actions within broader tactics.

Like ENA, ONA produces summary statistics for network comparison and mathematically consistent network visualizations that enable interpretation of statistical measures. Unlike ENA, ONA models the order in which codes appear in data, enabling researchers to investigate whether and to what extent the order of events is meaningful in a given context.

In the following sections, we provide a step-by-step guide to conducting ENA and ONA analyses in R.

3 Epistemic Network Analysis in R

In this section, we demonstrate how to conduct an ENA analysis using the rENA package. If you are not familiar with ENA as an analytic technique, we recommend that you first read Shaffer [1], Shaffer and Ruis [14], and Bowman et al. [2] to familiarize yourself with the theoretical and methodological foundations of ENA.

3.1 Install the rENA Package and Load the Library

Before installing the rENA package, be sure that you are using R version 4.1 or newer. To check your R version, type R.version in your console. To update your R version (if needed), download and install R from the official R website: https://cran.r-project.org/

First, install the rENA package and then load the rENA library after installation is complete.

```
install.packages("rENA", repos = c("https://cran.qe-libs.org", "https://cran.
rstudio.org"))

library(rENA)
```

We also install the other package that is required for accessing the view() function Sect. 3.7.3 in rENA.

```
install.packages("tma", repos = c("https://cran.qe-libs.org", "https://cran.r
studio.org"))

library(tma )
```

3.2 Dataset

The dataset we will use as an example, RS.data, is included in the rENA package. Note that the RS.data file in the package is only a subset of the full dataset, and is thus intended for demonstration purposes only.

To start, pass RS.data from the rENA package to a data frame named **data**.

```
data = rENA::RS.data
```

Use the head() function in R to subset and preview the first three rows present in the input data frame to familiarize yourself with the data structure.

```
head(data,3)
```

```
##      UserName Condition CONFIDENCE.Pre CONFIDENCE.Post CONFIDENCE.Change
## 1     steven z FirstGame              7               8                 1
## 2      akash v FirstGame              6               8                 2
## 3 alexander b FirstGame               5               7                 1
##   C.Level.Pre NewC.Change    C.Change      Timestamp ActivityNumber GroupNa
me
## 1    High.Pre   Pos.Change Pos.Change 9/17/2013 9:43              1  Electr
ic
## 2    High.Pre   Pos.Change Pos.Change 9/17/2013 9:44              1  Electr
ic
## 3     Low.Pre   Pos.Change Pos.Change 9/17/2013 9:46              1  Electr
ic
##   GameHalf GameDay             text Data Technical.Constraints
## 1    First       1           Steven    0                     0
## 2    First       1 Hey, I am Akash     0                     0
## 3    First       1        I'm Alex     0                     0
##   Performance.Parameters Client.and.Consultant.Requests Design.Reasoning
## 1                      0                              0                0
## 2                      0                              0                0
## 3                      0                              0                0
##   Collaboration
## 1             0
## 2             0
## 3             0
```

RS.data consists of discourse from *RescuShell*, an online learning simulation where students work as interns at a fictitious company to solve a realistic engineering design problem in a simulated work environment. Throughout the internship, students communicate with their project teams and mentors via online chat, and these chats are recorded in the “text” column. A set of qualitative codes were applied to the data in the “text” column, where a value of 0 indicates the absence of the code and a value of 1 indicates the presence of the code in a given line.

Further details about the RS.data dataset can be found in Shaffer and Arastoopour [15]. Analyses of data from *RescuShell* and other engineering virtual internships can be found in Arastoopour et al. [16] and Chesler et al. [17].

3.3 Construct an ENA Model

To construct an ENA model, there is a function called ena which enables researchers to set the parameters for their model. This function wraps two other functions—ena.accumulate.data and ena.make.set—which can be used together to achieve the same result.

In the following sections, we will demonstrate how to set each parameter and explain how different choices affect the resulting ENA model.

3.3.1 Specify Units

In ENA, *units* can be individuals, ideas, organizations, or any other entity whose structure of connections you want to model. To set the units parameter, specify which column(s) in the data contain the variables that identify unique units.

For this example, choose the “Condition” column and the “UserName” column to define the units. The “Condition” column has two unique values: FirstGame, and SecondGame, representing novice users and relative expert users, respectively, as some students participated in *RescuShell* after having already completed a different engineering virtual internship. The “UserName” column includes unique user names for all students (*n* = 48). This way of defining the units means that ENA will construct a network for each student in each condition.

```
unitCols = c("Condition", "UserName")
```

To verify that the units are correctly specified, subset and preview the unique values in the units columns. There are 48 units from two conditions, which means that the ENA model will produce 48 individual-level networks for each of the units, and each unit is uniquely associated with either the novice group (FirstGame) or the relative expert group (SecondGame).

```
head(unique(data[, unitCols]),3)
```

```
##    Condition     UserName
## 1  FirstGame    steven  z
## 2  FirstGame     akash  v
## 3 FirstGame alexander b
```

3.3.2 Specify Codes

Next, specify the columns that contain the *codes*. Codes are concepts whose pattern of association you want to model for each unit. ENA represent codes as nodes in the networks and co-occurrences of codes as edges. Most researchers use binary coding in ENA analyses, where the values in the code columns are either 0 (indicating that the code is not present in that line) or 1 (indicating that the code is present in that line). RS.data contains six code columns, all of which will be used here.

To specify the code columns, enter the code column names in a vector.

```
codeCols = c('Data', 'Technical.Constraints', 'Performance.Parameters', 'Clie
nt.and.Consultant.Requests', 'Design.Reasoning', 'Collaboration')
```

To verify that the codes are correctly specified, preview the code columns selected.

```
head(data[,codeCols],3)
```

```
##   Data Technical.Constraints Performance.Parameters
## 1    0                     0                      0
## 2    0                     0                      0
## 3    0                     0                      0
##   Client.and.Consultant.Requests Design.Reasoning Collaboration
## 1                              0                0             0
## 2                              0                0             0
## 3                              0                0             0
```

3.3.3 Specify Conversations

The conversation parameter determines which lines in the data *can* be connected. Codes in lines that are not in the same conversation cannot be connected. For example, you may want to model connections within different time segments, such as days, or different steps in a process, such as activities.

In our example, choose the “Condition”, “GroupName”, and “ActivityNumber” columns to define the conversations. These choices indicate that connections can only happen between students who were in the same condition (FirstGame or SecondGame) and on the same project team (group), and within the same activity. This definition of conversation reflects what actually happened in the simulation: in a given condition, students only interacted with those who were in the same group, and each activity occurred on a different day.

To specify the conversation parameter, enter the column names in a vector.

```
conversationCols = c("Condition", "GroupName", "ActivityNumber")
```

To verify that the conversations are correctly specified, subset and preview the unique values in the conversation columns.

```
head(unique(data[, conversationCols]),3)
```

```
##    Condition GroupName ActivityNumber
## 1  FirstGame  Electric              1
## 12 FirstGame  Electric              3
## 15 FirstGame  Electric              4
```

3.3.4 Specify the Window

Once the conversation parameter is specified, a window method needs to be specified. Whereas the conversation parameter specifies which lines *can be* related, the window parameter determines which lines within the same conversation *are* related. The most common window method used in ENA is called a moving stanza window, which is what will be used here.

Briefly, a moving stanza window is a sliding window of fixed length that moves through a conversation to detect and accumulate code co-occurrences in recent temporal context. The lines within a designated stanza window are considered related to each other. For instance, if the moving stanza window is 7, then each line in the conversation is linked to the six preceding lines. See Siebert-Evenstone et al. [18] and Ruis et al. [19] for more detailed explanations of windows in ENA models.

Here, set the window.size.back[1]parameter equal to 7. User can specify a different moving stanza window size by passing a different numerical value to the ‘window.size.back‘ parameter.

```
window.size.back = 7
```

3.3.5 Specify Groups and Rotation Method

When specifying the units, we chose a column that indicates two conditions: FirstGame (novice group) and SecondGame (relative expert group). To enable comparison of students in these two conditions, three additional parameters need to be specified: groupVar, groups, and mean.

[1] The ENA package also enables use of an infinite stanza window. The infinite stanza window works the same way as a moving stanza window, but there is no limit on the number of previous lines that are included in the window besides the conversation itself. The infinite stanza window is less commonly used in ENA, but is specified as follows: window.size.back = “INF”.

```
groupVar = "Condition" # "Condition" is the column used as our grouping varia
ble
groups = c("FirstGame", "SecondGame") # "FirstGame" and "SecondGame" are the
two unique values of the "Condition" column
mean = TRUE
```

These three parameters indicate that when building the ENA model, the first dimension will maximize the difference between the two conditions: FirstGame and SecondGame. This difference maximization is achieved through mean = TRUE, which specifies that a *means rotation* will be performed at the dimensional reduction stage. If the means rotation is set to FALSE or there aren't two distinct groups in your data and you still set mean as TRUE, ENA will by default use singular value decomposition (SVD) to perform the dimensional reduction. Bowman et al. [2] provide a mathematical explanation of the methods used in ENA to perform dimensional reductions.

3.3.6 Specify Metadata

The last parameter to be specified is metadata. Metadata columns are not required to construct an ENA model, but they provide information that can be used to subset units in the resulting model.

Specify the metadata columns shown below to include data on student outcomes related to reported self-confidence before and after participating in engineering virtual internships. We will use this data to demonstrate a simple linear regression analysis that can be done using ENA outputs as predictors.

```
metaCols = c("CONFIDENCE.Change","CONFIDENCE.Pre","CONFIDENCE.Post","C.Change
") # optional
```

3.3.7 Construct an Model

Now that all the essential parameters have been specified, the ENA model can be constructed.

The ena function constructs the ENA model, and we recommend that you store the output in an object (in this case, **set.ena).**

```
set.ena =
  ena(
    data = data,
    units = unitCols,
    codes = codeCols,
    conversation = conversationCols,
    window.size.back = 7,
    metadata = metaCols, # optional
    groupVar = groupVar,
    groups = groups,
    mean = TRUE
    )
```

As noted above, the ena helper function combines the functions ena.accumu late.data and ena.make.set. The following code will construct the same ENA model specified above using these two functions.

```
accum.ena =
  ena.accumulate.data(
    text_data = RS.data[, 'text'],
    units = data[,unitCols],
    conversation = data[,conversationCols],
    metadata = data[,metaCols], # optional
    codes = data[,codeCols],
```

```
    window.size.back = 7
)

set.ena =
  ena.make.set(
    enadata = accum.ena, # the accumulation ran above
    rotation.by = ena.rotate.by.mean, # equivalent of mean=TRUE in the ena fu
nction
    rotation.params = list(
      accum.ena$meta.data$Condition=="FirstGame", # equivalent of groups in t
he ena function
      accum.ena$meta.data$Condition=="SecondGame" # equivalent of groups in t
he ena function
  )
)
```

3.4 *Summary of Key Model Outputs*

Users can explore what is stored in the object **set** by typing set$ and select items from the drop down list. Here, we briefly describe the top-level items in set that are often of interest.

3.4.1 Connection Counts

Connection counts are the frequencies of unique connections a unit made. For each unit, ENA creates a cumulative adjacency vector that contains the sums of all unique code co-occurrences for that unit across all stanza windows. Here, there are 48 units in the ENA model, so there are 48 adjacency vectors. Each term in an ENA adjacency vector represents a unique co-occurrence of codes. Thus with six codes, each vector has 15 terms (*n* choose two). This is because ENA models are undirected and do not model co-occurrences of the same code.

To access ENA adjacency vectors, use set.ena$connection.counts.

```
head(set.ena$connection.counts,3)
```

```
##                 ENA_UNIT Condition     UserName CONFIDENCE.Change CONFIDENC
E.Pre
## 1:    FirstGame.steven z FirstGame     steven z                 1
7
## 2:     FirstGame.akash v FirstGame      akash v                 2
6
## 3: FirstGame.alexander b FirstGame alexander b                 1
5
##    CONFIDENCE.Post    C.Change Data & Technical.Constraints
## 1:               8 Pos.Change                            22
## 2:               8 Pos.Change                            47
## 3:               7 Pos.Change                             9
##    Data & Performance.Parameters Technical.Constraints & Performance.Param
eters
## 1:                            18
20
## 2:                            34
42
## 3:                             5
8
##    Data & Client.and.Consultant.Requests
## 1:                                     5
## 2:                                    10
## 3:                                     5
##    Technical.Constraints & Client.and.Consultant.Requests
## 1:                                                      6
## 2:                                                     14
## 3:                                                      3
##    Performance.Parameters & Client.and.Consultant.Requests
## 1:                                                       5
## 2:                                                      13
## 3:                                                       3
##    Data & Design.Reasoning Technical.Constraints & Design.Reasoning
## 1:                      21                                       26
## 2:                      45                                       59
## 3:                       5                                        8
##    Performance.Parameters & Design.Reasoning
## 1:                                        19
## 2:                                        38
## 3:                                         5
##    Client.and.Consultant.Requests & Design.Reasoning Data & Collaboration
## 1:                                                 6                    7
## 2:                                                 9                   12
## 3:                                                 2                    4
##    Technical.Constraints & Collaboration Performance.Parameters & Collabor
ation
## 1:                                     9
7
## 2:                                    21
11
## 3:                                     6
4
##    Client.and.Consultant.Requests & Collaboration
## 1: 1
## 2:                                              2
## 3:                                              0
##    Design.Reasoning & Collaboration
## 1: 6
## 2:                               19
## 3:                                5
```

3.4.2 Line Weights

To compare networks in terms of their *relative* patterns of association, researchers can spherically normalize the cumulative adjacency vectors by diving each one by its length. The resulting normalized vectors represent each unit's relative frequencies of code co-occurrence. In other words, the sphere normalization controls for the fact that different units might have different amounts of interaction or different numbers of activities than others.

Notice that in set.ena$connection.counts, the value for each unique code co-occurrence is an integer equal or greater than 0, because they represent the raw connection counts between each unique pair of codes. In set."ena"$line.weights, those raw counts are normalized, and therefore the values are rational numbers between 0 and 1.

To access the normalized adjacency vectors, use set.ena$line.weights.

```
head(set.ena$line.weights,3)
```

```
##                ENA_UNIT Condition     UserName CONFIDENCE.Change CONFIDENC
E.Pre
## 1:    FirstGame.steven z FirstGame     steven z                 1
7
## 2:     FirstGame.akash v FirstGame      akash v                 2
6
## 3: FirstGame.alexander b FirstGame alexander b                 1
5
##    CONFIDENCE.Post    C.Change Data & Technical.Constraints
## 1:               8 Pos.Change                    0.4000661
## 2:               8 Pos.Change                    0.4016067
## 3:               7 Pos.Change                    0.4370786
##    Data & Performance.Parameters Technical.Constraints & Performance.Param
eters
## 1:                     0.3273268                                      0.36
36965
## 2:                     0.2905240                                      0.35
88826
## 3:                     0.2428215                                      0.38
85143
##    Data & Client.and.Consultant.Requests
## 1:                            0.09092412
## 2:                            0.08544824
## 3:                            0.24282147
##    Technical.Constraints & Client.and.Consultant.Requests
## 1:                                              0.1091089
## 2:                                              0.1196275
## 3:                                              0.1456929
##    Performance.Parameters & Client.and.Consultant.Requests
## 1:                                              0.09092412
## 2:                                              0.11108271
## 3:                                              0.14569288
##    Data & Design.Reasoning Technical.Constraints & Design.Reasoning
## 1:                0.3818813                                0.4728054
## 2:                0.3845171                                0.5041446
## 3:                0.2428215                                0.3885143
##    Performance.Parameters & Design.Reasoning
## 1:                                 0.3455117
## 2:                                 0.3247033
## 3:                                 0.2428215
##    Client.and.Consultant.Requests & Design.Reasoning Data & Collaboration
## 1:                                        0.10910895            0.1272938
## 2:                                        0.07690342            0.1025379
## 3:                                        0.09712859            0.1942572
##    Technical.Constraints & Collaboration Performance.Parameters & Collabor
ation
## 1:                             0.1636634                                0.127
29377
## 2:                             0.1794413                                0.093
99306
## 3:                             0.2913858                                0.194
25717
##    Client.and.Consultant.Requests  &  Collaboration
## 1:                                      0.01818482
## 2:                                      0.01708965
## 3:                                      0.00000000
##    Design.Reasoning & Collaboration
## 1:                        0.1091089
## 2:                        0.1623517
## 3:                        0.2428215
```

3.4.3 ENA Points

As the product of a dimensional reduction, for each unit, ENA produces an ENA point in a two-dimensional space. Since there are 48 units, ENA produces 48 ENA points.

By default, rENA visualizes ENA points on an x-y coordinate plane defined by the first two dimensions of the dimensional reduction: for a means rotation, MR1 and SVD2, and for an SVD, SVD1 and SVD2.

To access these points, use set.ena$points.

```
head(set.ena$points,3)
```

```
##                  ENA_UNIT Condition     UserName CONFIDENCE.Change CONFIDENC
E.Pre
## 1:     FirstGame.steven z FirstGame     steven z                 1
7
## 2:      FirstGame.akash v FirstGame      akash v                 2
6
## 3: FirstGame.alexander b FirstGame alexander b                 1
5
##    CONFIDENCE.Post    C.Change          MR1         SVD2         SVD3
## 1:               8 Pos.Change -0.05423338  -0.008491458  0.06551249
## 2:               8 Pos.Change -0.07742095   0.031134440  0.03362490
## 3:               7 Pos.Change -0.30594927  -0.098348499 -0.01105519
##            SVD4          SVD5         SVD6         SVD7        SVD8        
SVD9
## 1:  2.034477e-02  0.011885463 -0.02427483  -0.023161244 -0.01643227 -0.0128
85771
## 2: -2.531589e-05  0.006465571  0.01336324   0.001215593 -0.01390223  0.0040
71313
## 3:  9.816549e-02 -0.003662261  0.07609149   0.077059745 -0.09198643 -0.0624
49678
##         SVD10        SVD11        SVD12        SVD13       SVD14        SV
D15
## 1: -0.01169155  0.005448827 -0.027880312 -0.006140204   0.01230336  0.01989
532
## 2: -0.04017379  0.035710843 -0.011559722  0.002192745   0.02967617 -0.01100
021
## 3: -0.01146149 -0.031675014  0.003382794  0.026665665  -0.01779259 -0.01083
445
```

ENA points are thus summary statistics that researchers can use to conduct statistical tests, and they can also be used in subsequent analyses. For example, statistical differences between groups in the data can be tested using ENA dimension scores, and those scores can also be used in regression analyses to predict outcome variables, which we will demonstrate later.

3.4.4 Rotation Matrix

The rotation matrix used during the dimensional reduction can be accessed through set.ena$rotation. This is mostly useful when you want to construct an ENA metric space using one dataset and then project ENA points from different data into that space, as in Sect. 5.1.

```
head(set.ena$rotation.matrix,3)
```

```
##                                                  codes          MR1         SVD
2
## 1:                  Data & Technical.Constraints -0.297140113  0.25542803
7
## 2:                 Data & Performance.Parameters  0.146745148 -0.40786334
0
## 3: Technical.Constraints & Performance.Parameters  0.006251295 -0.00672433
8
##          SVD3       SVD4        SVD5        SVD6       SVD7        SVD8
## 1:  0.4027380 0.1829314 -0.18841712 -0.22238612  0.3539426 -0.04618008
## 2:  0.4998255 0.1655853 -0.01271575  0.02831357 -0.5461929 -0.19336181
## 3: -0.0357799 0.4901334  0.29999562  0.27301947  0.1197999  0.45958816
##          SVD9       SVD10        SVD11         SVD12       SVD13       SVD14
## 1: 0.39593034  0.3556577 -0.03738872 -0.0005144122 0.30025177 0.24188117
## 2: 0.27237864 -0.1917366  0.18450254 -0.1247520324 0.17795795 0.06161493
 ## 3: 0.01614337 -0.2375881 -0.25630008 -0.3668342035 0.05404011 0.25435787
 ##          SVD15
 ## 1: 0.07161604
 ## 2: 0.04655934
 ## 3: 0.20907031
```

3.4.5 Metadata

set$meta.data returns a data frame that includes all the columns of the ENA set except for the columns representing code co-occurrences.

```
head(set.ena$meta.data,3)
```

```
##                  ENA_UNIT  Condition   UserName  CONFIDENCE.Change  CONFIDENC
E.Pre
## 1:    FirstGame.steven z  FirstGame     steven z                 1
7
## 2:     FirstGame.akash v  FirstGame      akash v                 2
6
## 3: FirstGame.alexander b  FirstGame alexander b                 1
5
##     CONFIDENCE.Post   C.Change
## 1:                8 Pos.Change
## 2:                8 Pos.Change
## 3:                7 Pos.Change
```

3.5 *ENA Visualization*

Once an ENA set is constructed, it can be visualized, which facilitates interpretation of the model. Here, we will look at the two conditions, “FirstGame” (novices) and “SecondGame” (relative experts), by plotting their mean networks.

3.5.1 Plot a Mean Network

To plot a network, use the ena.plot.network function. This function requires the network parameter (a character vector of line weights), and the line weights come from set$line.weights.

First, subset line weights for each of the two groups.

```
# Subset lineweights for FirstGame
first.game.lineweights = as.matrix(set.ena$line.weights$Condition$FirstGame)

# Subset lineweights for SecondGame
second.game.lineweights = as.matrix(set.ena$line.weights$Condition$SecondGame)
```

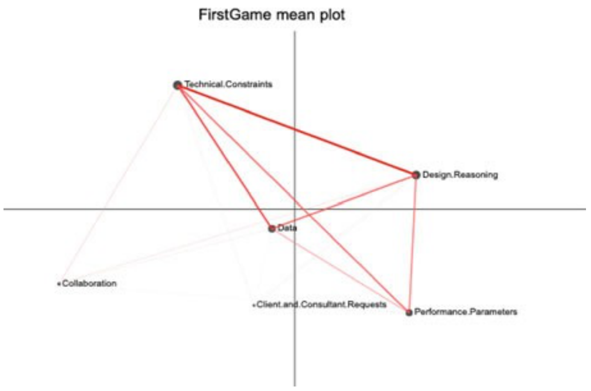


Fig. 1 ENA mean network for FirstGame group

Next, calculate the mean networks for the two groups, and store the line weights as vectors.

```
first.game.mean = as.vector(colMeans(first.game.lineweights))
second.game.mean = as.vector(colMeans(second.game.lineweights))
```

During plotting, use a pipe | > to send the output of one function into the first parameter of the subsequent function. To distinguish the two mean networks, set the color of the FirstGame mean network to red (Fig. 1).

```
ena.plot(set.ena, title = "FirstGame mean plot") |>
  ena.plot.network(network = first.game.mean, colors = c("red"))
```

and the color of the SecondGame mean network to blue (Fig. 2).

```
ena.plot(set.ena, title = "SecondGame mean plot") |>
  ena.plot.network(network = second.game.mean, colors = c("blue"))
```

As you can see from the two network visualizations above, their node positions are exactly same. All ENA networks from the same model have the same node positions, which are determined by an optimization routine that attempts to place the nodes such that the centroid of each unit's network and the location of the ENA point in the reduced space are co-located.

Because of the fixed node positions, ENA can construct a subtracted network, which enables the identification of the most salient differences between two networks. To do this, ENA subtracts the weight of each connection in one network from the corresponding weighted connection in another network, then visualizes the differences in connection strengths. Each edge is color-coded to indicate which of

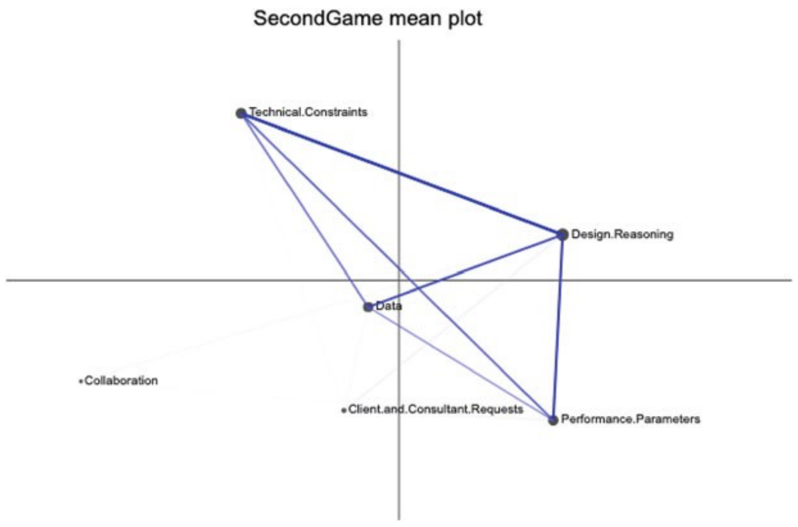


Fig. 2 ENA mean network for SecondGame group

the two networks contains the stronger connection, and the thickness and saturation of the edges corresponds to the magnitude of the difference.

To plot a subtracted network, first calculate the subtracted network line weights by subtracting one group's line weights from the other. (Because ENA computes the absolute values of the differences in edge weights, the order of the two networks in the subtraction doesn't matter.)

```
subtracted.mean = first.game.mean - second.game.mean
```

Then, use the ena.plot function to plot the subtracted network. If the differences are relatively small, a multiplier can be applied to rescale the line weights, improving legibility (Fig. 3).

```
ena.plot(set.ena, title = "Subtracted: FirstGame (red) - SecondGame (blue)")
|>
  ena.plot.network(network = subtracted.mean * 5, # Optional rescaling of the
line weights
                   colors = c("red", "blue"))
```

Here, the subtracted network shows that on average, students in the FirstGame condition (red) made more connections with Technical.Constraints and Collaboration than students in the SecondGame condition (blue), while students in the SecondGame condition made more connections with Design.Reasoning and Performance.Parameters than students in the FirstGame condition. This is because

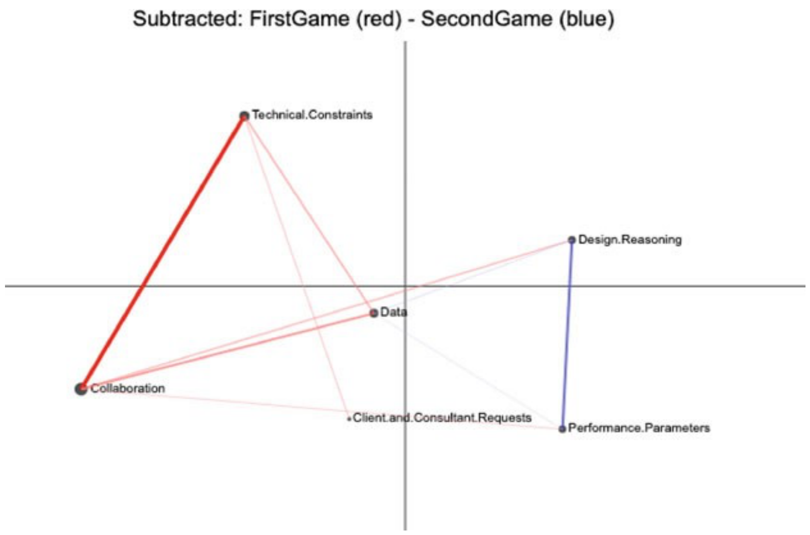

Subtracted: FirstGame (red) - SecondGame (blue)
Technical.Constraints
Design.Reasoning
Data
Collaboration
Client.and.Consultant.Requests
Performance.Parameters

Fig. 3 ENA subtracted mean network for FirstGame (red) and SecondGame (blue)

students with more experience of engineering design practices did not need to spend as much time and effort managing the collaborative process and learning about the basic technical elements of the problem space, and instead spent relatively more time focusing on more complex analysis and design reasoning tasks.

Note that this subtracted network shows no connection between Technical.Constraints and Design.Reasoning, simply because the strength of this connection was similar in both conditions. Thus, subtraction networks should always be visualized along with with the two networks being subtracted.

3.5.2 Plot a Mean Network and its Points

The ENA point or points associated with a network or mean network can also be visualized.

To visualize the points associated with each of the mean networks plotted above, use set$points to subset the rows that are in each condition and plot each condition as a different color.

```
# Subset rotated points for the first condition
first.game.points = as.matrix(set.ena$points$Condition$FirstGame)

# Subset rotated points for the second condition
second.game.points = as.matrix(set.ena$points$Condition$SecondGame)
```

Then, plot the FirstGame mean network the same as above using ena.plot.network, use | > to pipe in the FirstGame points that we want to include, and plot them using ena.plot.points.

Each point in the space is the ENA point for a given unit. The red and blue squares on the x-axis are the means of the ENA points for each condition, along with the 95% confidence interval on each dimension. (might need to zoom in for better readability).

Since we used a means rotation to construct the ENA model, the resulting space highlights the differences between FirstGame and SecondGame by constructing a rotation that places the means of each condition as close as possible to the x-axis of the space and maximizes the distance between them (Figs. 4, 5, 6, 7, and 8).

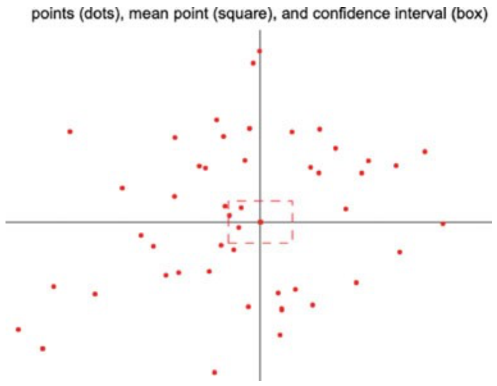


Fig. 4 FirstGame ENA points (dots), mean point (square), and confidence interval (box)

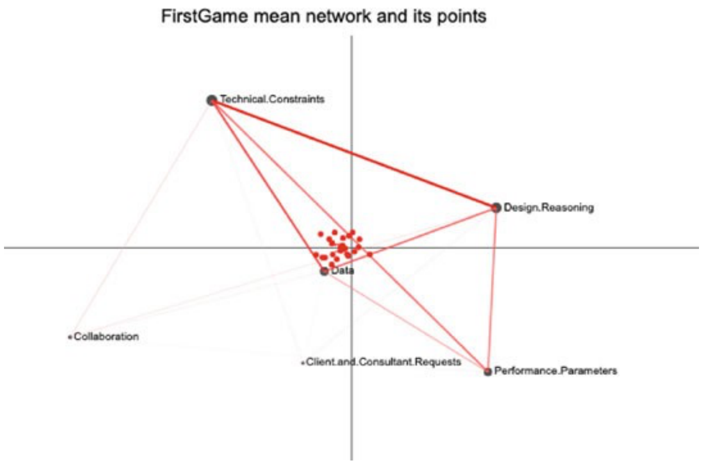


Fig. 5 FirstGame ENA mean network and points

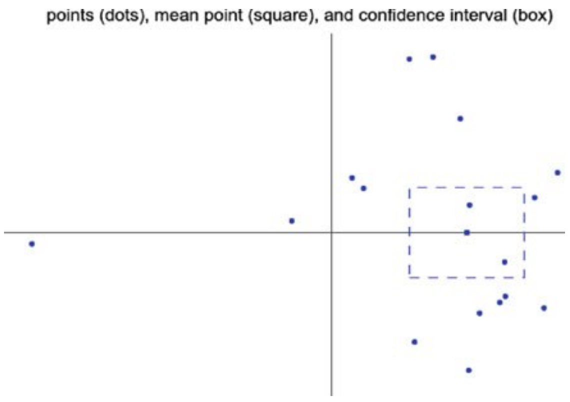


Fig. 6 SecondGame ENA points (dots), mean point (square), and confidence interval (box)

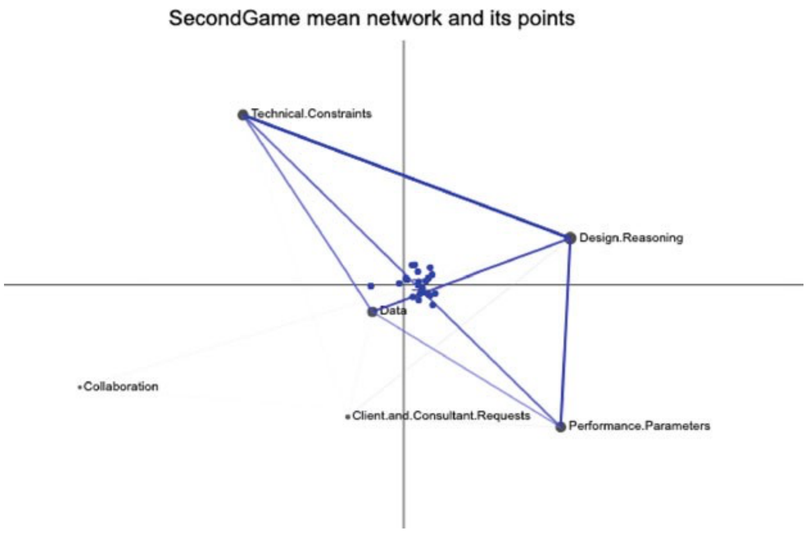


Fig. 7 SecondGame ENA mean network and points

```
ena.plot(set.ena, title = " points (dots), mean point (square), and confidenc
e interval (box)") |>
          ena.plot.points(points = first.game.points, colors = c("red")) |>
          ena.plot.group(point = first.game.points, colors =c("red"),
                         confidence.interval = "box")
```

```
ena.plot(set.ena, title = "FirstGame mean network and its points") |>
          ena.plot.network(network = first.game.mean, colors = c("red")) |>
          ena.plot.points(points = first.game.points, colors = c("red")) |>
          ena.plot.group(point = first.game.points, colors =c("red"),
                         confidence.interval = "box")
```

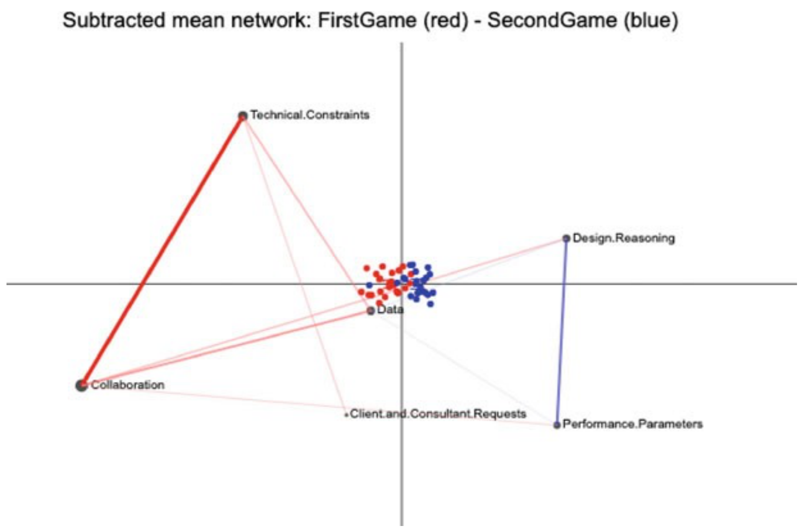


Fig. 8 ENA subtracted mean network for FirstGame (blue) and SecondGame (red)

Then, do the same for the SecondGame condition.

```
ena.plot(set.ena, title = " points (dots), mean point (square), and confidenc
e interval (box)") |>
          ena.plot.points(points = second.game.points, colors = c("blue")) |>
          ena.plot.group(point = second.game.points, colors =c("blue"),
                         confidence.interval = "box")
```

```
ena.plot(set.ena, title = "SecondGame mean network and its points") |>
          ena.plot.network(network = second.game.mean, colors = c("blue")) |>
          ena.plot.points(points = second.game.points, colors = c("blue")) |>
          ena.plot.group(point = second.game.points, colors =c("blue"),
                         confidence.interval = "box")
```

Lastly, do the same for subtraction as well.

```
ena.plot(set.ena, title = "Subtracted mean network: FirstGame (red) - SecondG
ame (blue)")  |>
          ena.plot.network(network = subtracted.mean * 5,
          colors = c("red", "blue")) |>
          ena.plot.points(points = first.game.points, colors = c("red")) |>
          ena.plot.group(point = first.game.points, colors =c("red"),
                         confidence.interval = "box") |>
          ena.plot.points(points = second.game.points, colors = c("blue")) |>
          ena.plot.group(point = second.game.points, colors =c("blue"),
                         confidence.interval = "box")
```

Note that the majority of the red points (FirstGame) are located on the left side of the space, and the blue points (SecondGame) are mostly located on the right side of the space. This is consistent with the line weights distribution in the mean network: the FirstGame units make relatively more connections with nodes on the left side of the space, while the SecondGame units make relatively more connections with nodes on the right side of the space. The positions of the nodes enable interpretation of the dimensions, and thus interpretation of the locations of the ENA points.

3.5.3 Plot an Individual Unit Network and its Point

Plotting the network and ENA point for a single unit uses the same approach. First, subset the line weights and point for a given unit.

```
unit.A.line.weights = as.matrix(set.ena$line.weights$ENA_UNIT$`FirstGame.stev
en z`) # subset line weights
unit.A.point = as.matrix(set.ena$points$ENA_UNIT$`FirstGame.steven z`) # subs
et ENA point
```

Then, plot the network and point for that unit (Fig. 9).

```
ena.plot(set.ena, title = "Individual network: FirstGame.steven z") |>
          ena.plot.network(network = unit.A.line.weights, colors = c("red"))
|>
          ena.plot.points(points = unit.A.point, colors = c("red"))
```

Following the exact same procedure, we can, for example, choose a unit from the other condition to plot and also construct a subtracted plot for those two units (Fig. 10).

```
unit.B.line.weights = as.matrix(set.ena$line.weights$ENA_UNIT$`SecondGame.sam
uel o`) # subset line weights
unit.B.point = as.matrix(set.ena$points$ENA_UNIT$`SecondGame.samuel o`) # sub
set ENA point

ena.plot(set.ena, title = "Individual network: SecondGame.samuel o") |>
          ena.plot.network(network = unit.B.line.weights, colors = c("blue"))
|>
          ena.plot.points(points = unit.B.point, colors = c("blue"))
```

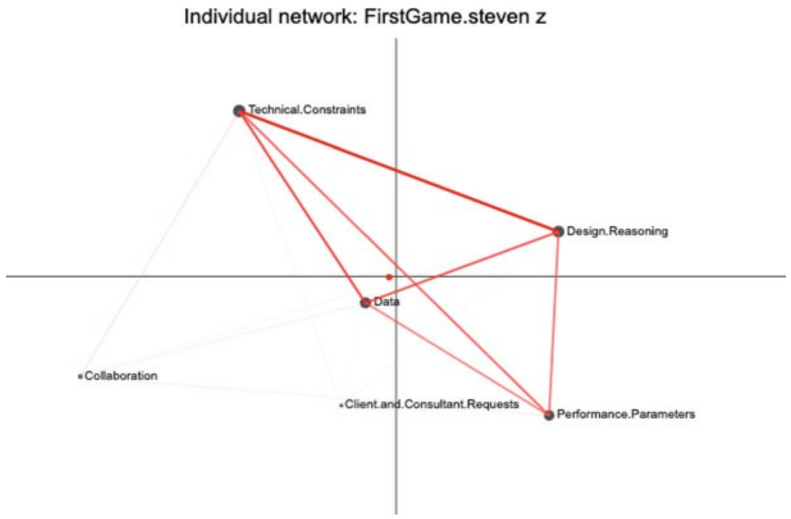


Fig. 9 EAN network for a student from FirstGame and its corresponding ENA point

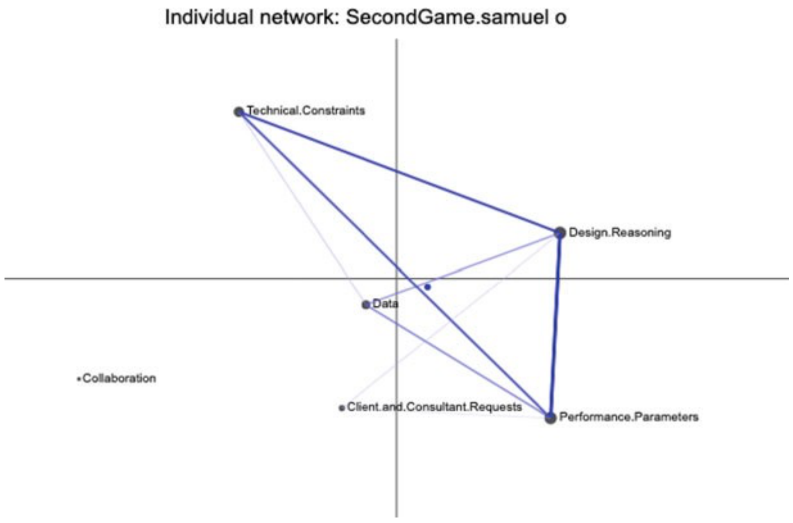


Fig. 10 ENA network for a student from SecondGame and its corresponding ENA point

To visually analyze the differences between the two individual networks, plot their subtracted network (Fig. 11).

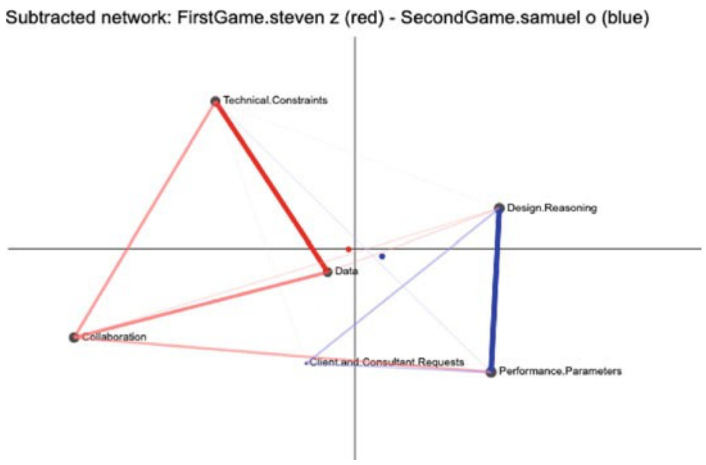

Subtracted network: FirstGame.steven z (red) - SecondGame.samuel o (blue)
Technical.Constraints
Design.Reasoning
Data
Collaboration
Client.and.Consultant.Requests
Performance.Parameters

Fig. 11 ENA subtracted network showing the differences between one student from FirstGame (red) and another student from SecondGame (blue)

```
ena.plot(set.ena, title = "Subtracted network: FirstGame.steven z (red) - Sec
ondGame.samuel o (blue)") |>
          ena.plot.network(network = (unit.A.line.weights - unit.B.line.weigh
ts) * 5,
          colors = c("red", "blue")) |>
          ena.plot.points(points = unit.A.point, colors = c("red")) |>
          ena.plot.points(points = unit.B.point, colors = c("blue"))
```

In this unit-level subtracted network, Unit A (red) made relatively more connections with codes such as Technical.Constraints, Data, and Collaboration, while Unit B (blue) made relatively more connections with Design.Reasoning and Performance.Parameters.

3.5.4 Plot Everything, Everywhere, All at Once

The helper function ena.plotter enables users to plot points, means, and networks for each condition at the same time. This gives the same results as above more parsimoniously. However, this approach does not enable customization of edge and point colors.

```
#with helper function
plot = ena.plotter(set.ena,
                      points = T,
                      mean = T,
                      network = T,
                      print.plots = T,
                      groupVar = "Condition",
                      groups = c("SecondGame","FirstGame"),
                      subtractionMultiplier = 5)
```

3.6 Compare Groups Statistically

In addition to visual comparison of networks, ENA points can be analyzed statistically. For example, here we might test whether the patterns of association in one condition are significantly different from those in the other condition.

To demonstrate both parametric and non-parametric approaches to this question, the examples below use a Student's *t* test and a Mann-Whitney *U* test to test for differences between the FirstGame and SecondGame condition. For more on differences between parametric and non-parametric tests, see Kaur and Kumar [20].

First, install the lsr package to enable calculation of effect size (Cohen's *d*) for the *t* test.

```
install.packages('lsr')
library(lsr)
```

Then, subset the points to test for differences between the points of the two conditions.

```
ena_first_points_d1 = as.matrix(set.ena$points$Condition$FirstGame)[,1]
ena_second_points_d1 = as.matrix(set.ena$points$Condition$SecondGame)[,1]

ena_first_points_d2 = as.matrix(set.ena$points$Condition$FirstGame)[,2]
ena_second_points_d2 = as.matrix(set.ena$points$Condition$SecondGame)[,2]
```

Conduct the *t* test on the first and second dimensions.

```
# parametric tests
t_test_d1 = t.test(ena_first_points_d1, ena_second_points_d1)
t_test_d1
```

```
##
##  Welch Two Sample t-test
##
## data:  ena_first_points_d1 and ena_second_points_d1
## t = -6.5183, df = 45.309, p-value = 5.144e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.2687818 -0.1419056
## sample estimates:
##   mean of x   mean of y
## -0.09411588  0.11122786
```

```
t_test_d2 = t.test(ena_first_points_d2, ena_second_points_d2)
t_test_d2
```

```
##
##  Welch Two Sample t-test
##
## data:  ena_first_points_d2 and ena_second_points_d2
## t = 1.9334e-16, df = 43.175, p-value = 1
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.07768526  0.07768526
## sample estimates:
##     mean of x     mean of y
##  1.935145e-19 -7.254914e-18
```

Compute any other statistics that may be of interest. A few examples are given below.

```
mean(ena_first_points_d1)
```

```
## [1] -0.09411588
```

```
mean(ena_second_points_d1)
```

```
## [1] 0.1112279
```

```
mean(ena_first_points_d2)
```

```
## [1] 1.935145e-19
```

```
mean(ena_second_points_d2)
```

```
## [1] -7.254914e-18
```

```
sd(ena_first_points_d1)
```

```
## [1] 0.1115173
```

```
sd(ena_second_points_d1)
```

```
## [1] 0.1063515
```

```
sd(ena_first_points_d2)
```

```
## [1] 0.1267104
```

```
sd(ena_second_points_d2)
```

```
## [1] 0.1380851
```

```
length(ena_first_points_d1)
```

```
## [1] 26
```

```
length(ena_second_points_d1)
```

```
## [1] 22
```

```
length(ena_first_points_d2)
```

```
## [1] 26
```

```
length(ena_second_points_d2)
```

```
## [1] 22
```

```
cohensD(ena_first_points_d1, ena_second_points_d1)
```

```
## [1] 1.880622
```

```
cohensD(ena_first_points_d2, ena_second_points_d2)
```

```
## [1] 5.641688e-17
```

Here, along the x axis (MR1), a two-sample t test assuming unequal variance shows that the FirstGame (mean = −0.09, SD = 0.11, N = 26) condition is statistically significantly different for alpha = 0.05 from the SecondGame condition (mean = 0.11, SD = 0.10, N = 22; t(45.31) = −6.52, p = 0.00, Cohen's d = 1.88). Along the y axis (SVD2), a two-sample t test assuming unequal variance shows that the FirstGame condition (mean = 0.11, SD = 0.13, N = 26) is not statistically significantly different for alpha = 0.05 from the SecondGame condition (mean = 0.00, SD = 1.3, N = 22; t(43.17) = 0, p = 1.00).

The Mann-Whitney U test is a non-parametric alternative to the independent two-sample t test.

First, install the rcompanion package to calculate the effect size (r) for a Mann-Whitney U test.

```
install.packages('rcompanion')
library(rcompanion)
```

Then, conduct a Mann-Whitney U test on the first and second dimensions.

```
# non parametric tests
w_test_d1 = wilcox.test(ena_first_points_d1, ena_second_points_d1)
w_test_d2  = wilcox.test(ena_first_points_d2,  ena_second_points_d2)

w_test_d1
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  ena_first_points_d1 and ena_second_points_d1
## W = 50, p-value = 8.788e-08
## alternative hypothesis: true location shift is not equal to 0
```

```
w_test_d2
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  ena_first_points_d2 and ena_second_points_d2
## W = 287, p-value = 0.9918
## alternative hypothesis: true location shift is not equal to 0
```

Compute any other statistics that may be of interest. A few examples are given below.

```
median(ena_first_points_d1)
```

```
## [1] -0.08464154
```

```
median(ena_second_points_d1)
```

```
## [1] 0.1300029
```

```
median(ena_first_points_d2)
```

```
## [1] -0.007252397
```

```
median(ena_second_points_d2)
```

```
## [1] 0.0003031848
```

```
length(ena_first_points_d1)
```

```
## [1] 26
```

```
length(ena_second_points_d1)
```

```
## [1] 22
```

```
length(ena_first_points_d2)
```

```
## [1] 26
```

```
length(ena_second_points_d2)
```

```
## [1] 22
```

```
abs(wilcoxonR(ena_first_points_d1, ena_second_points_d1))
```

```
##     r
## 0.863
```

```
abs(wilcoxonR(ena_first_points_d2, ena_second_points_d2))
```

```
##     r
## 0.863
```

Here, along the *x* axis (MR1), a Mann-Whitney *U* test shows that the FirstGame condition (Mdn = −0.08, N = 26) was statistically significantly different for alpha = 0.05 from the SecondGame condition (Mdn = −0.007, N = 22; *U* = 50, *p* = 0.00, *r* = 0.86). Along the *y* axis (SVD2), a Mann-Whitney *U* test shows that the FirstGame condition (Mdn = 0.13, N = 26) is not statistically significantly different for alpha = 0.05 from the SecondGame condition (Mdn = 0.00, N = 22; U = 287, *p* = 0.99). The absolute value of r value in Mann-Whitney U test varies from 0 to close to 1. The interpretation values for r commonly in published literature is: 0.10 - < 0.3 (small effect), 0.30 - < 0.5 (moderate effect) and > = 0.5 (large effect).

3.7 Model Evaluation

In this section, we introduce three ways users can evaluate the quality of their ENA models.

3.7.1 Variance Explained

Briefly, variance explained (also called explained variation) refers to the proportion of the total variance in a dataset that is accounted for by a statistical model or set of predictors.

In ENA, to represent high-dimensional vectors in a two-dimensional space, ENA uses either singular value decomposition or means rotation combined with SVD. For each of the reduced dimensions, the variance in patterns of association among units explained by that dimension can be computed.

```
head(set.ena$model$variance,2)
```

```
##       MR1      SVD2
## 0.3204602 0.2445006
```

Here, the first dimension is MR1 and the second dimension is SVD2. The MR1 dimension has the highest variance explained at 32%.

As with any statstical model, greater explained variance does not necessarily indicate a better model, as it may be due to overfitting, but it provides one indicator of model quality.

3.7.2 Goodness of Fit

Briefly, a model's goodness of fit refers to how well a model fits or represents the data. A model with a high goodness of fit indicates that it accurately represents the data and can make reliable predictions.

In ENA, a good fit means that the positions of the nodes in the space—and thus the network visualizations—are consistent with the mathematical properties of the model. In other words, we can confidently rely on the network visualizations to interpret the ENA model. The process that ENA uses to achieve high goodness of fit is called co-registration. The mathematical details of co-registration are beyond the scope of this chapter and can be found in Bowman et al. [2].

To test a model's goodness of fit, use ena.correlations. The closer the value is to 1, the higher the model's goodness of fit is. Most ENA models have a goodness of fit that is well above 0.90.

```
ena.correlations(set.ena)
```

```
##     pearson  spearman
## 1  0.993766  0.994119
## 2 0.9850392 0.9850519
```

3.7.3 Close the Interpretative Loop

Another approach to evaluate an ENA model is to confirm the alignment between quantitative model (in our case, our ENA model) and the original qualitative data. In other words, we can return to the original data to confirm that quantitative findings give a fair representation of the data. This approach is an example of what's called as closing the interpretative loop in Quantitative Ethnography field [1].

For example, based on our visual analysis of the network of "SecondGame.samuel o" in previous section, we are interested in what the lines are in the original data that contributed to the connection between Design.Reasoning and Performance.Parameters.

Let's first review what "SecondGame.samuel o" ENA network looks like (Fig. 12).

```
ena.plot(set.ena, title = "Individual network: SecondGame.samuel o") |>
          ena.plot.network(network = as.matrix(set.ena$line.weights$ENA_UNIT$
`SecondGame.samuel o`), colors = c("blue")) |>
          ena.plot.points(points = as.matrix(set.ena$points$ENA_UNIT$`SecondG
ame.samuel o`), colors = c("blue"))
```

To do so, we use view() function and specify required parameters as below.

This is going to activate a window shows up in your Viewer panel. If it is too small to read, you can click on the "Show in new window" button to view it in your browser for better readability.

In the Viewer panel, hover over your cursor on any of the lines that are in bold, a size of 7 lines rectangle shows up, representing that in a moving stanza window of size 7, the referent line (the line in bold) and its preceding 6 lines. The 1 and 0 in Technical.Constraints column and Design.Reasoning column shows where the connections happened (Fig. 13).

For example, line 2477 Samuel shared his [Design.Reasoning] about "mindful of (the) how one device scores relative to other ones", to reference back to what Casey said in line 2476 about [Performance.Parameters] "not one source/censor can be the best in every area so we had to sacrifice certain attributes", as well as what Jackson said in line 2475 about safety as one of the [Performance.Parameters] "when it came to the different attributes, i think that all were important in their own way but i think safety is one of the most important".

This is a qualitative example of a connection made between Performance.Parameters and Design.Reasoning.

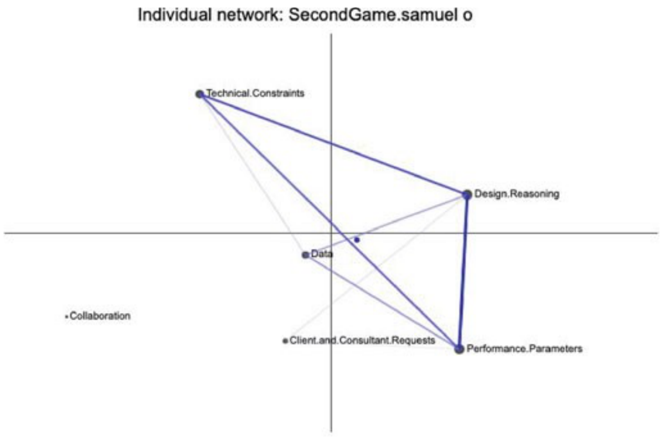
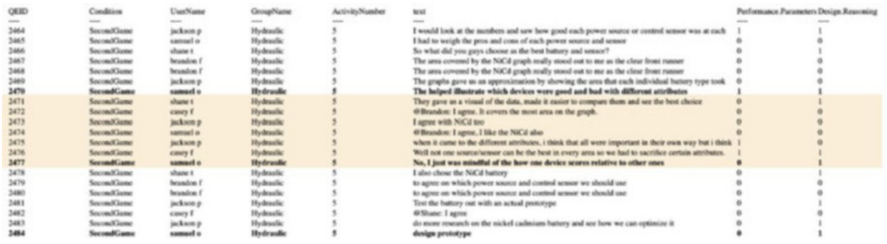


Fig. 12 ENA network for Samuel O from SecondGame



QEID	Condition	UserName	GroupName	ActivityNumber	text	Performance.Parameters	Design.Reasoning
2464	SecondGame	jackson p	Hydraulic	5	I would look at the numbers and saw how good each power source or control sensor was at each	1	1
2465	SecondGame	samuel o	Hydraulic	5	I had to weigh the pros and cons of each power source and sensor	0	0
2466	SecondGame	shane t	Hydraulic	5	So what did you guys choose as the best battery and sensor?	0	1
2467	SecondGame	brandon f	Hydraulic	5	The area covered by the NiCd graph really stood out to me as the clear front runner	0	0
2468	SecondGame	brandon f	Hydraulic	5	The area covered by the NiCd graph really stood out to me as the clear front runner	0	0
2469	SecondGame	jackson p	Hydraulic	5	The graphs gave us an approximation by showing the area that each individual battery type took	0	0
2470	**SecondGame**	**samuel o**	**Hydraulic**	**5**	**The helped illustrate which devices were good and bad with different attributes**	**1**	**1**
2471	SecondGame	shane t	Hydraulic	5	They gave us a visual of the data, made it easier to compare them and see the best choice	0	1
2472	SecondGame	casey f	Hydraulic	5	@Brandon: I agree. It covers the most area on the graph.	0	0
2473	SecondGame	jackson p	Hydraulic	5	I agree with NiCd too	0	0
2474	SecondGame	samuel o	Hydraulic	5	@Brandon: I agree, I like the NiCd also	0	0
2475	SecondGame	jackson p	Hydraulic	5	when it came to the different attributes, i think that all were important in their own way but i think	1	0
2476	SecondGame	casey f	Hydraulic	5	Well not one source/sensor can be the best in every area so we had to sacrifice certain attributes.	1	1
2477	**SecondGame**	**samuel o**	**Hydraulic**	**5**	**No, I just was mindful of the how one device scores relative to other ones**	**0**	**1**
2478	SecondGame	shane t	Hydraulic	5	I also chose the NiCd battery	0	1
2479	SecondGame	brandon f	Hydraulic	5	to agree on which power source and control sensor we should use	0	0
2480	SecondGame	brandon f	Hydraulic	5	to agree on which power source and control sensor we should use	0	0
2481	SecondGame	jackson p	Hydraulic	5	Test the battery out with an actual prototype	0	1
2482	SecondGame	casey f	Hydraulic	5	@Shane: I agree	0	0
2483	SecondGame	jackson p	Hydraulic	5	do more research on the nickel cadmium battery and see how we can optimize it	0	1
2484	**SecondGame**	**samuel o**	**Hydraulic**	**5**	**design prototype**	**0**	**1**

Fig. 13 A screenshot of the view() function result. The highlighted lines represent lines within the same stanza window

3.8 *Using ENA Model Outputs in Other Analyses*

It is often useful to use the outputs of ENA models in subsequent analyses. The most commonly used outputs are the ENA points, i.e., set$points. For example, we can use a linear regression analysis to test whether ENA points on the first two dimensions are predictive of an outcome variable, in this case, change in confidence in engineering skills.

```
regression_data = set.ena$points
regression_data$CONFIDENCE.Change = as.numeric(regression_data$CONFIDENCE.Cha
nge)

## Warning: NAs introduced by coercion

condition_regression = lm(CONFIDENCE.Change ~ MR1 + SVD2 + Condition,
                          data = regression_data,
                          na.action = na.omit)
summary(condition_regression)
```

```
##
## Call:
## lm(formula = CONFIDENCE.Change ~ MR1 + SVD2 + Condition, data = regression
_data,
##     na.action = na.omit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max

## -1.18092 -0.24324 -0.08171  0.30716  1.88404
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.1111     0.1490   7.457 2.82e-09 ***
## MR1                  -0.4540     0.8616  -0.527    0.601
## SVD2                  0.3268     0.7154   0.457    0.650
## ConditionSecondGame  -0.3484     0.2566  -1.358    0.182
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6374 on 43 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.1228, Adjusted R-squared:  0.0616
## F-statistic: 2.007 on 3 and 43 DF,  p-value: 0.1273
```

The results of this regression analysis show that ENA points are not a significant predictor of the students' pre-post change in confidence (MR1: $t = -0.53, p = 0.60$; SVD2: $t = 0.46, p = 0.65$; Condition: $t = -1.36, p = 0.18$). The overall model was also not significant (F(3, 43) $= 2.01, p = 0.13$) with an adjusted r-squared value of 0.06.

Recall that the dataset we are using is a small subset of the full RS.data, and thus results that are significant for the whole dataset may not be for this sample.

4 Ordered Network Analysis with R

This section demonstrates how to conduct an ONA analysis using the ona R package. If you are new to ONA as an analytic technique, Tan et al. [12] provides a more detailed explication of its theoretical and methodological foundations.

Because ONA shares some conceptual and procedural similarities with ENA, you may also want to read the recommended papers from the ENA section [1, 2, 14].

4.1 Install the ONA Package and Load the Library

Install the ona package and load the ona library after installing.

```
install.packages("ona", repos = c("https://cran.qe-libs.org",
"https://cran.rstudio.org"))

library(ona)
```

Then, install the other package that is required for ONA analysis.

```
install.packages("tma", repos = c("https://cran.qe-libs.org",
"https://cran.rstudio.org"))

library(tma)
```

4.2 Dataset

(Refer to Sect. 3.2 for a detailed description of the dataset used here.)

Load the RS.data dataset.

```
data = ona::RS.data
```

4.3 Construct an ONA Model

To construct an ONA model, identify which columns in the data to use for the parameters required by the ONA modeling function. The parameters are defined identically in both ENA and ONA; see Sect. 3.3 for detailed explanations.

4.3.1 Specify Units

Select the *units* as in Sect. 3.3.1.

```
my_units <- c("Condition", "UserName")
```

4.3.2 Specify Codes

Select the *codes* as in Sect. 3.3.2.

```
my_codes = c(
           'Data',
           'Technical.Constraints',
           'Performance.Parameters',
           'Client.and.Consultant.Requests',
           'Design.Reasoning',
           'Collaboration')
```

4.3.3 Specify Conversations

The parameter to specify *conversations* in rENA is called “conversation”; in ONA, the equivalent is called “my_hoo_rules”, where “hoo” is an abbreviation of “horizon of observation.”

Choose the combination of “Condition” column, “GroupName” column, and “ActivityNumber” column to define the conversation parameter.

The syntax to specify conversations using my_hoo_rules in ONA is slightly different from the syntax to specify conversation in ENA, but the conceptual definition is the same.

```
my_hoo_rules <- conversation_rules(
                  (Condition  %in%  UNIT$Condition  &
                   GroupName %in% UNIT$GroupName &
                   ActivityNumber %in% UNIT$ActivityNumber))
```

4.3.4 Specify the Window

Specify a moving stanza window size by passing a numerical value to the window_size parameter.

```
window_size = 7
```

4.3.5 Specify Metadata

As in ENA, metadata columns can be included if desired. Metadata columns are not required to construct an ONA model, but they provide information that can be used to subset units in the resulting model.

```
metaCols = c("CONFIDENCE.Change","CONFIDENCE.Pre","CONFIDENCE.Post","C.Change
")
```

4.3.6 Accumulate Connections

Now that all the parameters are specified, connections can be accumulated. For each unit, the ONA algorithm uses a moving stanza window to identify connections formed from a current line of data (e.g., a turn of talk), or *response*, to the preceding lines within the window (the *ground*).

Unlike in ENA, where connections among codes are recorded in a symmetric adjacency matrix, ONA accounts for the order in which the connections occur by constructing an *asymmetric adjacency matrix* for each unit; that is, the number of connections from code A to code B may be different than the number of connections from B to A.

To accumulate connections, pass the parameters specified to the contexts and accumulate_contexts functions, and store the output in an object (in this case, **accum.ona).**

```
accum.ona <-
  contexts(data,
           units_by = my_units,
           hoo_rules = my_hoo_rules) |>
  accumulate_contexts(codes = my_codes,
                      decay.function = decay(simple_window, window_size = 7),
                      meta.data = metaCols,
                      return.ena.set = FALSE) # keep this as FALSE to get an
ONA model, otherwise it will return an undirected model)
```

4.3.7 Construct an ONA Model

After accumulation, call the model function to construct an ONA model. ONA currently implements *singular value decomposition* (SVD) and *means rotation* (MR) to perform dimensional reduction.

To create an ONA model using SVD, pass the accum.ona object to the model function.

```
set.ona <-
  model(accum.ona)
```

When there are two discrete groups to compare, a means rotation can be used, as described in Sect. 3.3.5.

A means rotation is specified using rotate.using ="mean" in the model function. Additionally, the model function expects rotation.params to be a list with two named elements, each containing a logical vector representing the rows of units to be included in each group.

```
set.ona <-
  model(accum.ona,                     # the previously run accumulation above
        rotate.using ="mean",          # means rotation method
        rotation.params =              # two groups for means rotation in a list
              list(FirstGame=accum.ona$meta.data$Condition=="FirstGame",
                   SecondGame=accum.ona$meta.data$Condition=="SecondGame")
        )
```

Here, construct the ONA model as shown below.

4.4 *Summary of Key Model Outputs*

Information about an ONA model is now stored in the R object set.ona.

As in rENA, users can explore the data stored in the object by typing set.ona$ and select items from the drop down list. Here, we briefly explain the top-level items in set.ona$.

4.4.1 Connection Counts

Because ONA accounts for the order in which the connections occur by constructing an *asymmetric adjacency matrix* for each unit, connection counts *from* code A *to* code B and *from* B *to* A, as well as self-connections for each code (from A to A) are recorded. Thus, because six codes were included in the model, the cumulative adjacency vector for each unit contains 36 terms (*n*ˆ2).

```
head(set.ona$connection.counts,3)
```

```
##    Condition     UserName                ENA_UNIT Data to Data
## 1: FirstGame     steven z     FirstGame::steven z           26
## 2: FirstGame      akash v      FirstGame::akash v           72
## 3: FirstGame alexander b FirstGame::alexander b           11
##    Technical.Constraints to Data Performance.Parameters to Data
## 1:                          44.0                             32
## 2:                         102.5                             66
## 3:                          21.5                             15
##     Client.and.Consultant.Requests to Data Design.Reasoning to Data
## 1:                                      12                        52
## 2:                                      10                        88
## 3:                                    4                       14
##    Collaboration to Data Data to Technical.Constraints
## 1:                   8                         27.0
## 2:                  12                        106.5
## 3:                   1                         11.5
##    Technical.Constraints to Technical.Constraints
## 1: 63
## 2:                                          193
## 3:                                           40
##    Performance.Parameters to Technical.Constraints
## 1: 29.5
## 2:                                          90.5
## 3:                                          15.5
##    Client.and.Consultant.Requests to Technical.Constraints
## 1: 8.5
## 2:                                                  23.5
## 3:                                                   1.0
##    Design.Reasoning to Technical.Constraints
## 1: 62.0
## 2:                                   160.5
## 3:                                    23.0
##    Collaboration to Technical.Constraints Data to Performance.Parameters
## 1:                                 11.0                           24
## 2:                                 29.0                           77
## 3:                                 15.5                           10
##    Technical.Constraints to Performance.Parameters
## 1: 35.5
## 2:                                          91.5
## 3:                                          17.5
##    Performance.Parameters to Performance.Parameters
## 1: 34
## 2:                                             72
## 3:                                             10
##    Client.and.Consultant.Requests to Performance.Parameters
## 1: 5.5
## 2:                                                   20.5
## 3:                                                    3.0
##    Design.Reasoning to Performance.Parameters
## 1: 42
## 2:                                      77
## 3:                                      14
##    Collaboration to Performance.Parameters Data to Client.and.Consultant.R
equests
## 1:                                   7.5
6
## 2:                                  14.5
18
## 3:                                   1.0
5
##    Technical.Constraints to Client.and.Consultant.Requests
## 1:                                                     7.5
## 2:                                                    19.5
## 3:                                                     7.0
##    Performance.Parameters to Client.and.Consultant.Requests
## 1:                                                      11.5
## 2:                                                      18.5
## 3:                                                       3.0
##    Client.and.Consultant.Requests  to  Client.and.Consultant.Requests
## 1:                                                                 5
## 2:                                                                12
## 3:                                                                 1
##     Design.Reasoning to Client.and.Consultant.Requests
## 1:                                                15.5
## 2:                                                 9.0
## 3:                                                 3.0
##    Collaboration to Client.and.Consultant.Requests Data to Design.Reasonin
g
## 1:                                               2                      19
## 2:                                               3                      86
## 3:                                               0                      10
##    Technical.Constraints to Design.Reasoning
## 1:                                      50.0
## 2:                                     152.5
## 3:                                      17.0
##    Performance.Parameters to Design.Reasoning
## 1:                                        20
## 2:                                        69
## 3:                                         9
##    Client.and.Consultant.Requests  to  Design.Reasoning
## 1:                                                 5.5
## 2:                                                16.0
## 3:                                                 2.0
##    Design.Reasoning to Design.Reasoning Collaboration to Design.Reasoning
## 1:                                   59                               6.0
## 2:                                  136                              33.5
## 3:                                   19                               4.0
##    Data to Collaboration Technical.Constraints to Collaboration
## 1:                     0                                    7.0
## 2:                    15                                   27.0
## 3:                     6                                   18.5
##    Performance.Parameters to Collaboration
## 1:                                     0.5
## 2:                                     8.5
## 3:                                     6.0
##    Client.and.Consultant.Requests  to  Collaboration
## 1:                                                0
## 2:                                                2
## 3:                                                0
##    Design.Reasoning to Collaboration Collaboration to Collaboration
## 1:                               5.0                              0
## 2:                              42.5                             14
## 3:                               7.0                              9
```

4.4.2 Line Weights

To compare networks in terms of their *relative* patterns of association, researchers can spherically normalize the cumulative adjacency vectors by diving each one by its length. The resulting normalized vectors represent each unit's relative frequencies of code co-occurrence. In other words, the sphere normalization controls for the fact that different units might have different amounts of interaction or different numbers of activities than others.

In set.ona$connection.counts, the value for each unique co-occurrence of codes is an integer equal or greater than 0, because they represent the directional connection counts between each pair of codes. In set.ona$line.weights, the connection counts are sphere normalized, and so the values are between 0 and 1.

```
head(set.ona$line.weights,3)
```

```
##    Condition     UserName                ENA_UNIT ENA_DIRECTION Data to Data
## 1: FirstGame     steven z    FirstGame::steven z      response   0.1543564
## 2: FirstGame      akash v     FirstGame::akash v      response   0.1619657
## 3: FirstGame alexander b FirstGame::alexander b       response   0.1424314
##    Technical.Constraints to Data Performance.Parameters to Data
## 1:                     0.2612185                      0.1899771
## 2:                     0.2305762                      0.1484686
## 3:                     0.2783886                      0.1942246
##     Client.and.Consultant.Requests to Data Design.Reasoning to Data
## 1:                              0.07124140                0.3087127
## 2:                              0.02249524                0.1979581
## 3:                              0.05179323                0.1812763
##    Collaboration to Data Data to Technical.Constraints
## 1:            0.04749427                     0.1602931
## 2:            0.02699429                     0.2395743
## 3:            0.01294831                     0.1489055
##     Technical.Constraints to Technical.Constraints
## 1:                                       0.3740173
## 2:                                       0.4341581
## 3:                                       0.5179323
##    Performance.Parameters to Technical.Constraints
## 1:                                       0.1751351
## 2:                                       0.2035819
## 3:                                       0.2006988
##     Client.and.Consultant.Requests to Technical.Constraints
## 1:                                                0.05046266
## 2:                                                0.05286381
## 3:                                                0.01294831
##     Design.Reasoning to Technical.Constraints
## 1:                                  0.3680806
## 2:                                  0.3610486
## 3:                                0.2978111
##    Collaboration to Technical.Constraints Data to Performance.Parameters
## 1:                            0.06530462                        0.1424828
## 2:                            0.06523619                        0.1732133
## 3:                            0.20069875                        0.1294831
##    Technical.Constraints to Performance.Parameters
## 1:                                      0.2107558
## 2:                                      0.2058314
## 3:                                      0.2265954
##    Performance.Parameters to Performance.Parameters
## 1:                                       0.2018506
## 2:                                       0.1619657
## 3:                                       0.1294831
##    Client.and.Consultant.Requests  to  Performance.Parameters
## 1:                                              0.03265231
## 2:                                              0.04611524
## 3:                                              0.03884492
##     Design.Reasoning to Performance.Parameters
## 1:                                  0.2493449
## 2:                                  0.1732133
## 3:                                  0.1812763
##    Collaboration to Performance.Parameters Data to Client.and.Consultant.R
equests
## 1:                             0.04452587                            0.035
62070
## 2:                             0.03261810                            0.040
49143
## 3:                             0.01294831                            0.064
74153
##    Technical.Constraints to Client.and.Consultant.Requests
## 1: 0.04452587
## 2:                                             0.04386571
## 3:                                             0.09063815
##    Performance.Parameters to Client.and.Consultant.Requests
## 1:                                              0.06827301
## 2:                                              0.04161619
## 3:                                              0.03884492
##    Client.and.Consultant.Requests  to  Client.and.Consultant.Requests
## 1:                                                      0.02968392
## 2:                                                      0.02699429
## 3:                                                      0.01294831
##    Design.Reasoning to Client.and.Consultant.Requests
## 1:                                        0.09202014
## 2:                                        0.02024571
## 3:                                        0.03884492
##    Collaboration to Client.and.Consultant.Requests Data to Design.Reasonin
g
## 1:                                    0.011873567              0.1127989
## 2:                                    0.006748571              0.1934590
## 3:                                    0.000000000              0.1294831
##    Technical.Constraints to Design.Reasoning
## 1:                                 0.2968392
## 2:                                 0.3430524
## 3:                                 0.2201212
##    Performance.Parameters to Design.Reasoning
## 1:                                  0.1187357
## 2:                                  0.1552171
## 3:                                  0.1165348
##    Client.and.Consultant.Requests to Design.Reasoning
## 1:                                         0.03265231
## 2:                                         0.03599238
## 3:                                         0.02589661
##    Design.Reasoning to Design.Reasoning Collaboration to Design.Reasoning
## 1:                            0.3502702                        0.03562070
## 2:                            0.3059352                        0.07535905
## 3:                            0.2460178                        0.05179323
##    Data to Collaboration Technical.Constraints to Collaboration
## 1:            0.00000000                             0.04155748
## 2:            0.03374286                             0.06073714
## 3:            0.07768984                             0.23954367
##     Performance.Parameters to Collaboration
## 1:                              0.002968392
## 2:                              0.019120952
## 3:                              0.077689840
##    Client.and.Consultant.Requests to Collaboration
## 1:                                     0.000000000
## 2:                                     0.004499048
## 3:                                     0.000000000
##    Design.Reasoning to Collaboration Collaboration to Collaboration
## 1:                        0.02968392                     0.00000000
## 2:                        0.09560476                     0.03149333
## 3:                        0.09063815                     0.11653476
```

4.4.3 ONA Points

For each unit, ONA produces an ONA point in a two-dimensional space formed by the first two dimensions of the dimensional reduction.

Here, the MR1 column represents the *x*-axis coordinate for each unit, and the SVD2 column represents the *y*-axis coordinate for each unit.

```
head(set.ona$points,3)
```

```
##    Condition     UserName                ENA_UNIT  ENA_DIRECTION          MR1
## 1: FirstGame     steven z    FirstGame::steven z      response   0.00753635
## 2: FirstGame      akash v     FirstGame::akash v      response  -0.07719283
## 3: FirstGame alexander b FirstGame::alexander b      response -0.20600855
##           SVD2        SVD3        SVD4       SVD5         SVD6          SVD
7
## 1: -0.05350532  0.02308722  0.03365899 0.18576251 -0.064647347 -0.01638733
9
```

```
## 2:  0.01840812 -0.01485049 -0.03634380 0.02065882 -0.003406081 -0.00802670
5
## 3: -0.05806135 -0.08409658  0.13340227 0.03017168 -0.046102014 -0.09567464
9
##           SVD8        SVD9      SVD10         SVD11        SVD12        SVD
13
## 1:  0.02504138 -0.04662639 0.01654204 -0.0050339193  0.001089911  0.020150
19
## 2: -0.01584017 -0.01879391 0.03338419 -0.0082095228 -0.004596587  0.050346
82
## 3: -0.10337234   0.14247946 0.01749875  0.0005982879 -0.073284812 -0.019597
35
##          SVD14        SVD15       SVD16       SVD17         SVD18         S
VD19
## 1: -0.03170014   0.014028551 -0.03292583 -0.01286482  0.0002353795 -0.01426
1325
## 2: -0.02280942   0.042727200  0.02373320 -0.02829064 -0.0208970211  0.01012
0401
## 3: -0.01633710 -0.002697103 -0.03717024  0.02019079 -0.0119060565 -0.00651
8936
##          SVD20        SVD21        SVD22        SVD23        SVD24
## 1:  0.01159296 0.0009366777 -0.033851942 -0.003230414 -0.016553686
## 2: -0.01752303 0.0045814224 -0.006533689 -0.007668149  0.017903544
## 3: -0.03746771 0.0075254949  0.038768078  0.008012232  0.009589488
##            SVD25        SVD26        SVD27         SVD28        SVD29
## 1:  2.124235e-03  0.008351416  0.024600338 -0.0003563738  0.006848068
## 2: -4.643724e-05 -0.013146155 -0.003761067 -0.0172067645 -0.009072690
## 3:  1.347210e-03  0.014229380  0.006423058  0.0020076670  0.003970270
##           SVD30       SVD31       SVD32         SVD33        SVD34
SVD35
## 1:  0.009214306 0.000689622 0.001926662 -0.0001196152 0.0057286766  0.0031
16433
## 2: -0.006574454 0.001518995 0.006561024  0.0037200464 0.0019185441  0.0064
47044
## 3: -0.003266165 0.004738366 0.003057179  0.0018318401 0.0008428842 -0.0048
84616
##           SVD36
## 1:  0.003361801
## 2: -0.001230173
## 3:  0.001080194
```

4.4.4 Rotation Matrix

The rotation matrix used during the dimensional reduction can be accessed through set.ona$rotation. This is mostly useful when you want to construct an ONA metric space using one dataset and then project ONA points from different data into that space, as in Sect. 5.2.

```
head(set.ona$rotation.matrix,3)
##                              codes         MR1       SVD2         SVD3
## 1:                    Data to Data -0.06930733 -0.4261566  0.041261221
## 2:  Technical.Constraints to Data -0.18483941 -0.3305414 -0.001705309
## 3: Performance.Parameters to Data  0.16511484 -0.4098705 -0.057056137
##            SVD4        SVD5       SVD6       SVD7        SVD8       SVD9
## 1: -0.24813254 -0.12386178 -0.1212851 -0.1813800 -0.02634818 0.02463562
## 2: -0.09652822  0.29776415 -0.2487902  0.4530700 -0.08131861 0.19027807
## 3: -0.06943836  0.02894105 -0.1666548 -0.1512543  0.26221091 0.15918103
##         SVD10      SVD11      SVD12       SVD13      SVD14       SVD15
## 1: -0.1685089 0.02779782  0.1634913  0.03515259 -0.1307011  0.11108864
## 2: -0.2985731 0.03356852 -0.1788196  0.24614774 -0.1162268 -0.06526278
## 3:  0.2634069 0.30265567  0.0364710 -0.16880948  0.2246206  0.04608908
##         SVD16       SVD17      SVD18        SVD19       SVD20       SVD21
## 1:  0.1678676 -0.21955326  0.4390146  0.263975884 -0.03791243 -0.26637411
## 2:  0.2427948  0.31464439 -0.1658976 -0.023955700 -0.04756904  0.05450570
## 3: -0.1521164 -0.06523085 -0.3436932  0.005934216 -0.21114098  0.07531336
##          SVD22       SVD23       SVD24       SVD25       SVD26       SVD27
## 1: -0.16378180 -0.04863544 -0.03046897  0.01129295  0.07990007 -0.10906748
## 2:  0.09542229  0.04913771  0.09391774 -0.07213795 -0.01391693 -0.03060512
## 3:  0.18550364 -0.20153993  0.11835360 -0.08759355 -0.06613629  0.01739627
##         SVD28       SVD29       SVD30       SVD31        SVD32        SVD3
3
## 1:  0.1218933  0.05135484  -0.08227500  0.09675121 -0.205485302 -0.15944684
3
## 2: -0.0431860 -0.04667436  -0.03855961 -0.06592640 -0.006227139 -0.00619161
0
## 3:  0.1294605  0.01342884 -0.17369350  0.07209455 -0.153340746 -0.00982495
5
##           SVD34       SVD35      SVD36
## 1:  -0.12907089  0.14685408 0.01703190
## 2:   0.02971639  0.11327562 0.07985307
## 3:   0.17078477 -0.02710075 0.06842069
```

4.4.5 Metadata

set.ona$meta.data gives a data frame that includes all the columns except for the code connection columns.

```
head(set.ona$meta.data,3)
```

```
##    Condition    UserName                ENA_UNIT
## 1: FirstGame    steven z    FirstGame::steven z
## 2: FirstGame     akash v     FirstGame::akash v
## 3: FirstGame alexander b FirstGame::alexander b
```

4.5 ONA Visualization

Once an ONA model is constructed, ONA networks can be visualize. The plotting function in ONA is called plot, and it works similarly to the same function in ENA.

Before plotting, you can set up several global parameters to ensure consistency across plots. These parameters will be clearer in subsequent sections.

```r
node_size_multiplier = 0.4 # scale up or down node sizes
node_position_multiplier = 1 # zoom in or out node positions
point_position_multiplier =1.5 # zoom in or out the point positions
edge_arrow_saturation_multiplier = 1.5 # adjust the chevron color lighter or
darker
edge_size_multiplier = 1 # scale up or down edge sizes
```

4.5.1 Plot a Mean Network

Mean ONA networks can be plotted for each of the conditions along with their subtracted network.

First, plot the mean network for the FirstGame condition. Use a pipe | > to connect the edges function and the nodes function. Users are only required to specify the weights parameter, as the remaining parameters have default values unless specified otherwise (Fig. 14).

```r
ona:::plot.ena.ordered.set(set.ona, title = "FirstGame (red) mean network") |
>
  edges(
    weights =set.ona$line.weights$Condition$FirstGame,
    edge_size_multiplier = edge_size_multiplier,
    edge_arrow_saturation_multiplier = edge_arrow_saturation_multiplier,
    node_position_multiplier = node_position_multiplier,
    edge_color = c("red")) |>
  nodes(
    node_size_multiplier = node_size_multiplier,
    node_position_multiplier = node_position_multiplier,
    self_connection_color = c("red"))
```

Since this is the first ONA network visualization in this chapter, we briefly explain how to read an ONA network.

Node size: In ONA, the node size is proportional to the number of occurrences of that code as a *response* to other codes in the data, with larger nodes indicating more responses. For example, in this plot, students in the FirstGame condition responded most frequently with discourse about Technical.Constraints.

Self-connections: The color and saturation of the circle within each node is proportional to the number of *self-connections* for that code: that is, when a code is both what students *responded to* and what they *responded with.* Colored circles that are larger and more saturated reflect codes with more frequent self-connections.

Edges: Note that unlike most directed network visualizations, which use arrows or spearheads to indicate direction, ONA uses a “broadcast” model, where the source of a connection (what students responded to) is placed at the apex side of the triangle and the destination of a connection (what students responded with) is placed at its base.

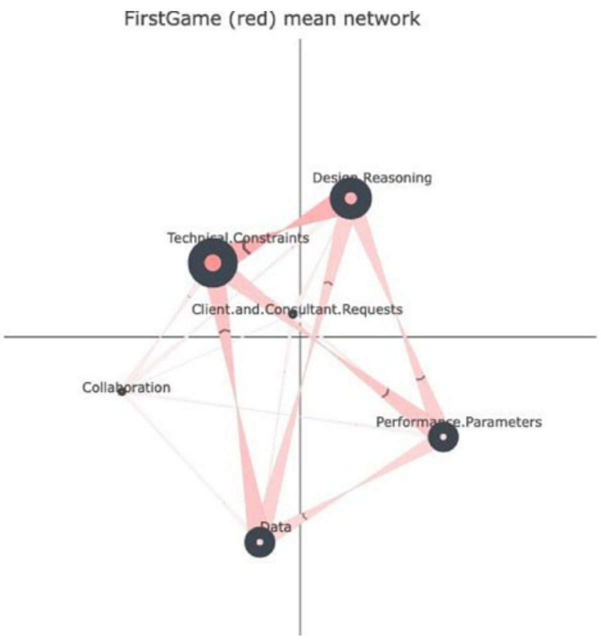


Fig. 14 ONA mean network for FirstGame group

Chevrons on edges: The chevrons point in the direction of the connection. Between any pair of nodes, if there is a bidirectional connection, the chevron only appears on the side with the stronger connection. This helps viewers differentiate heavier edges in cases such as between Technical.Constraints and Data, where the connection strengths from both directions are similar. When the connection strengths are identical between two codes, the chevron will appear on both edges.

Now, plot the mean network for SecondGame (Fig. 15).

```
ona:::plot.ena.ordered.set(set.ona, title = "SecondGame (blue) mean network")
|>
  edges(
    weights = set.ona$line.weights$Condition$SecondGame,
    edge_size_multiplier = edge_size_multiplier,
    edge_arrow_saturation_multiplier = edge_arrow_saturation_multiplier,
    node_position_multiplier = node_position_multiplier,
    edge_color = c("blue")) |>
  nodes(
    node_size_multiplier = node_size_multiplier,
    node_position_multiplier = node_position_multiplier,
    self_connection_color = c("blue"))
```

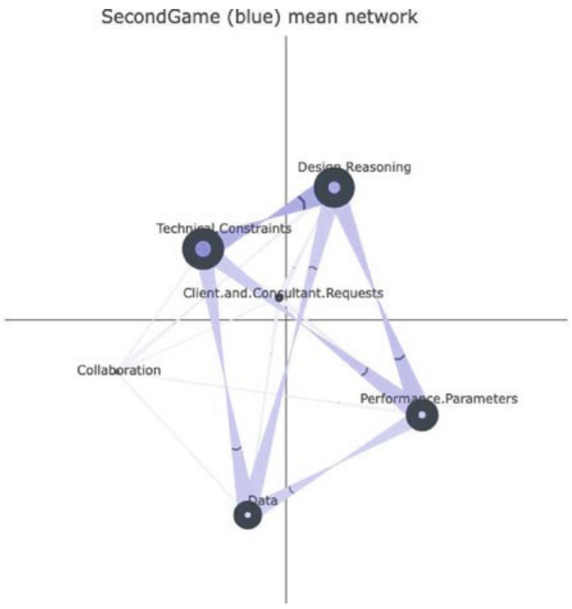


Fig. 15 ONA mean network for SecondGame group

Then, plot the subtracted network to show the differences between the mean networks of the FirstGame and SecondGame conditions (Fig. 16).

```
ona:::plot.ena.ordered.set(set.ona, title = "Subtracted mean network: FirstGa
me (red) vs SecondGame (blue)") |>
  edges(
    weights = (colMeans(set.ona$line.weights$Condition$FirstGame) - colMeans(
set.ona$line.weights$Condition$SecondGame))*4, # optional weights multiplier
to adjust readability
    edge_size_multiplier = edge_size_multiplier,
    edge_arrow_saturation_multiplier = edge_arrow_saturation_multiplier,
    node_position_multiplier = node_position_multiplier,
    edge_color = c("red","blue")) |>
  nodes(
    node_size_multiplier = node_size_multiplier,
    node_position_multiplier = node_position_multiplier,
    self_connection_color = c("red","blue"))
```

4.5.2 Plot a Mean Network and its Points

Besides plotting the mean network for each condition and the subtracted network, we can also plot the individual units within each condition.

Use set.ona$points to subset the rows that are in each condition and plot the units in each condition as a different color.

The points are specified in the units function. The edges and nodes functions remain the same as above (Figs. 17, 18, 19, and 20).

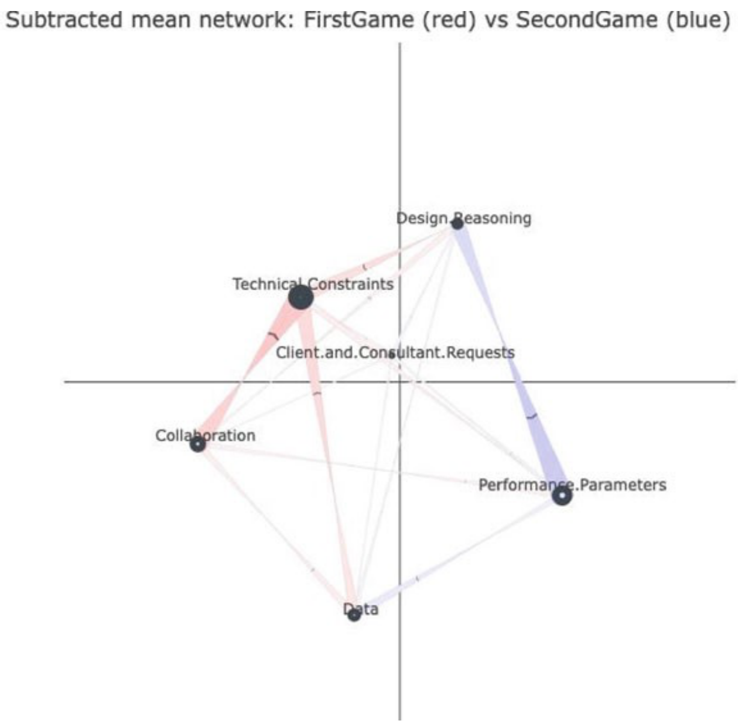


Fig. 16 ONA subtracted network showing the differences between FirstGame (red) and SecondGame (blue)

```
ona:::plot.ena.ordered.set(set.ona, title = "points (dots), mean point (squar
e), and confidence interval") |>
  units(
    points=set.ona$points$Condition$FirstGame,
    points_color = c("red"),
    show_mean = TRUE, show_points = TRUE, with_ci = TRUE)
```

```
ona:::plot.ena.ordered.set(set.ona, title = "FirstGame (red) mean network") |
>
  units(
    points=set.ona$points$Condition$FirstGame,
    points_color = c("red"),
    show_mean = TRUE, show_points = TRUE, with_ci = TRUE) |>
  edges(
    weights =set.ona$line.weights$Condition$FirstGame,
    edge_size_multiplier = edge_size_multiplier,
    edge_arrow_saturation_multiplier = edge_arrow_saturation_multiplier,
    node_position_multiplier = node_position_multiplier,
    edge_color = c("red")) |>
  nodes(
    node_size_multiplier = node_size_multiplier,
    node_position_multiplier = node_position_multiplier,
    self_connection_color = c("red"))
```

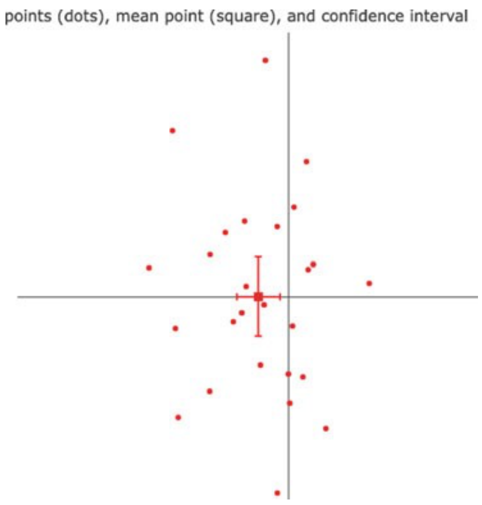


Fig. 17 ONA points (dots) and their mean point (square) for FirstGame group

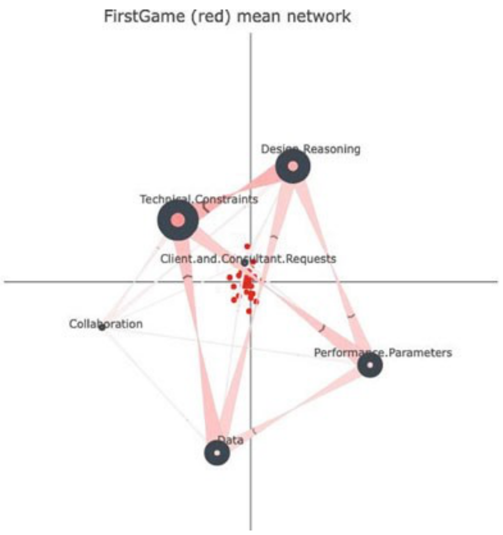


Fig. 18 ONA mean network for FirstGame group

```
ona:::plot.ena.ordered.set(set.ona, title = "points (dots), mean point (squar
e), and confidence interval") |>
  units(
    points=set.ona$points$Condition$SecondGame,
    points_color = c("blue"),
    show_mean = TRUE, show_points = TRUE, with_ci = TRUE)
```

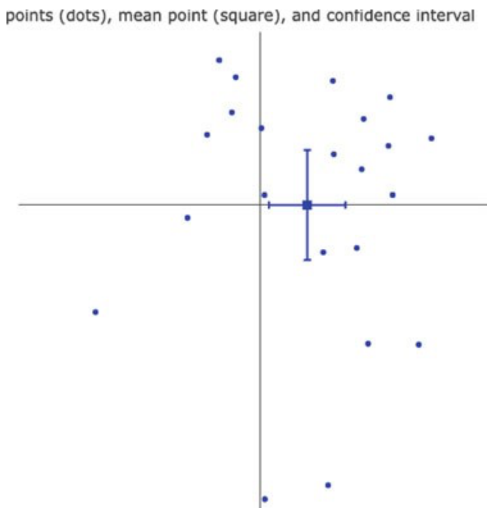


Fig. 19 ONA points (dots) and their mean point (square) for SecondGame group

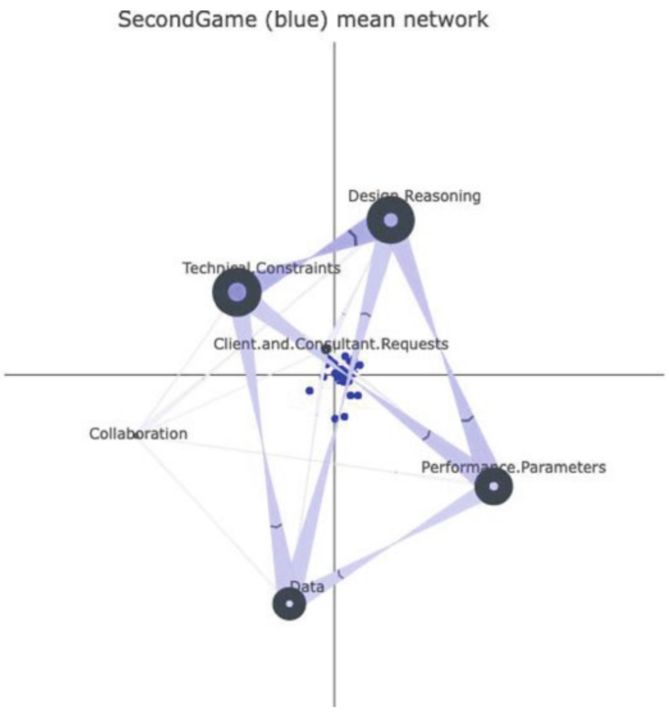


Fig. 20 ONA mean network and points for SecondGame

```
ona:::plot.ena.ordered.set(set.ona, title = "SecondGame (blue) mean network")
|>
  units(
    points=set.ona$points$Condition$SecondGame,
    points_color = "blue",
    show_mean = TRUE, show_points = TRUE, with_ci = TRUE) |>
  edges(
    weights = set.ona$line.weights$Condition$SecondGame,
    edge_size_multiplier = edge_size_multiplier,
    edge_arrow_saturation_multiplier = edge_arrow_saturation_multiplier,
    node_position_multiplier = node_position_multiplier,
    edge_color = c("blue")) |>
  nodes(
    node_size_multiplier = node_size_multiplier,
    node_position_multiplier = node_position_multiplier,
    self_connection_color = c("blue"))
```

Plot the subtracted network as follows (Fig. 21).

```
# FirstGame and SecondGame subtracted plot
ona:::plot.ena.ordered.set(set.ona, title = "Difference: FirstGame (red) vs S
econdGame (blue)") |>
  units(
    points = set.ona$points$Condition$FirstGame,
    points_color = "red",
    show_mean = TRUE, show_points = TRUE, with_ci = TRUE) |>
  units(
    points = set.ona$points$Condition$SecondGame,
    points_color = "blue",
    show_mean = TRUE, show_points = TRUE, with_ci = TRUE) |>
  edges(
    weights = (colMeans(set.ona$line.weights$Condition$FirstGame) - colMeans(
```

```
set.ona$line.weights$Condition$SecondGame))*4, # optional multiplier to adjus
t for readability
    edge_size_multiplier = edge_size_multiplier,
    edge_arrow_saturation_multiplier = edge_arrow_saturation_multiplier,
    node_position_multiplier = node_position_multiplier,
    edge_color = c("red","blue")) |>
  nodes(
    node_size_multiplier = node_size_multiplier,
    node_position_multiplier = node_position_multiplier,
    self_connection_color = c("red","blue"))
```

4.5.3 Plot an Individual Network and its Points

To plot an individual student's network and ONA point, use set.ona$points.

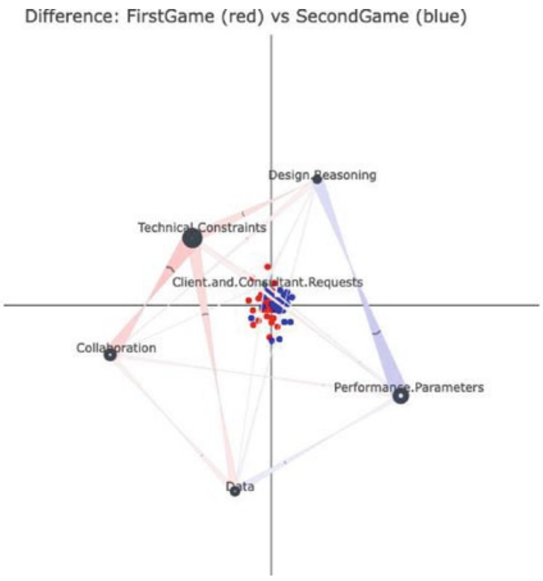


Fig. 21 ONA subtracted network showing the differences between FirstGame (red) and SecondGame (blue)

Here, we choose the same two units we compared in the ENA analysis (Sect. 3.5.3) (Figs. 22 and 23).

```
# first game
ona:::plot.ena.ordered.set(set.ona, title = "FirstGame::steven z") |>
  units(
    points=set.ona$points$ENA_UNIT$`FirstGame::steven z`,
    points_color = "red",
    show_mean = FALSE, show_points = TRUE, with_ci = FALSE) |>
  edges(
    weights = set.ona$line.weights$ENA_UNIT$`FirstGame::steven z`,
    edge_size_multiplier = edge_size_multiplier,
    edge_arrow_saturation_multiplier = edge_arrow_saturation_multiplier,
    node_position_multiplier = node_position_multiplier,
    edge_color = c("red")) |>
  nodes(
    node_size_multiplier = node_size_multiplier,
    node_position_multiplier = node_position_multiplier,
    self_connection_color = c("red"))
```

```
# second game
ona:::plot.ena.ordered.set(set.ona, title = "SecondGame::samuel o") |>
  units(
    points=set.ona$points$ENA_UNIT$`SecondGame::samuel o`,
    points_color = "blue",
    show_mean = FALSE, show_points = TRUE, with_ci = FALSE) |>
  edges(
    weights = set.ona$line.weights$ENA_UNIT$`SecondGame::samuel o`,
    edge_size_multiplier = edge_size_multiplier,
    edge_arrow_saturation_multiplier = edge_arrow_saturation_multiplier,
    node_position_multiplier = node_position_multiplier,
    edge_color = c("blue")) |>
  nodes(
    node_size_multiplier = node_size_multiplier,
    node_position_multiplier = node_position_multiplier,
    self_connection_color = c("blue"))
```

In this case, both units make relatively strong connections between Design.Reasoning and Data. However, for Unit A (red), the connection is relatively more *from* Design.Reasoning *to* Data than the other way around. This indicates that more often this unit responded with Data. In contrast, Unit B (blue) responded more frequently to Data with Design.Reasoning.

A subtracted network can make such differences more salient (Fig. 24).

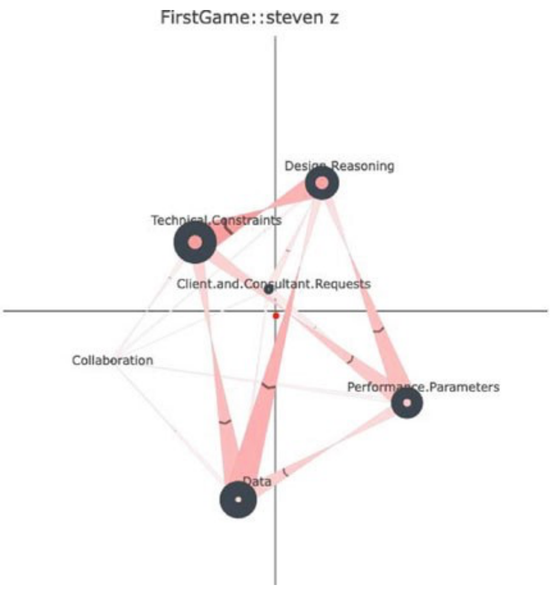


Fig. 22 ONA network for a student from FirstGame

```
# units difference
mean1 = as.vector(as.matrix(set.ona$line.weights$ENA_UNIT$`FirstGame::steven
z`))
mean2  =  as.vector(as.matrix(set.ona$line.weights$ENA_UNIT$`SecondGame::samuel
o`))

subtracted.mean = mean1 - mean2

ona:::plot.ena.ordered.set(set.ona, title = "subtracted network of steven z.F
irstGame.Electric and SecondGame.luke u") |>
  units(
    points = set.ona$points$ENA_UNIT$`FirstGame::steven z`, points_color = "r
ed",
    point_position_multiplier = point_position_multiplier,
    show_mean = FALSE, show_points = TRUE, with_ci = FALSE) |>
  units(
    points = set.ona$points$ENA_UNIT$`SecondGame::samuel o`, points_color = "
blue",
    point_position_multiplier = point_position_multiplier,
    show_mean = FALSE, show_points = TRUE, with_ci = FALSE) |>
  edges(
    weights = subtracted.mean*2,
    edge_size_multiplier = edge_size_multiplier,
    edge_arrow_saturation_multiplier = edge_arrow_saturation_multiplier,
    node_position_multiplier = node_position_multiplier,
    edge_color = c("red", "blue")) |>
  nodes(
    node_size_multiplier = node_size_multiplier,
    node_position_multiplier = node_position_multiplier,
    self_connection_color = c("red", "blue"))
```

The connection between Design.Reasoning and Data consists of two triangles, one in blue pointing from Data to Design.Reasoning, the other in red pointing from Design.Reasoning to Data. This indicates that although both units made strong connections between these two codes, the relative directed frequencies are different. Recall that in the ENA subtracted network for the same two units, the connections between Data and Design.Reasoning were basically the same. ONA, by accounting for the order of events, shows that while the undirected relative frequencies were similar, there was a difference in the order in which the two students made the connection.

4.6 Compare Groups Statistically

In addition to visual comparison of networks, ENA points can be analyzed statistically. For example, here we might test whether the patterns of association in one condition are significantly different from those in the other condition.

To demonstrate both parametric and non-parametric approaches to this question, the examples below use a Student's *t* test and a Mann-Whitney *U* test to test for differences between the FirstGame and SecondGame condition.

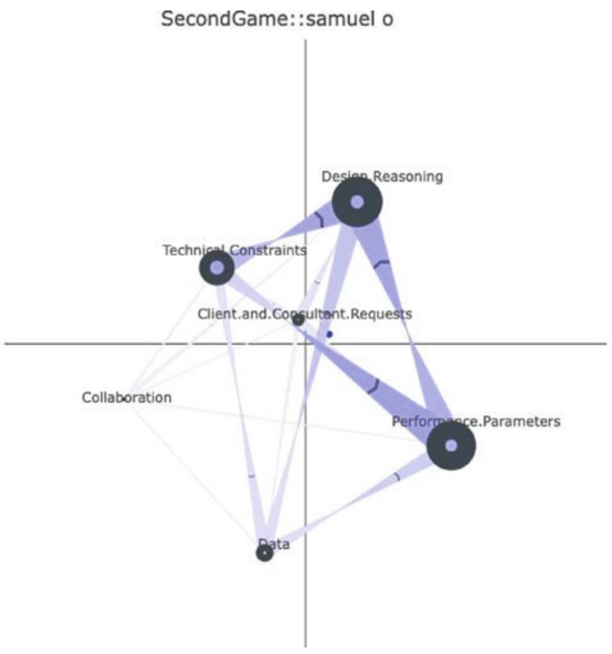


Fig. 23 ONA network for a student from SecondGame

First, install the lsr package to enable calculation of effect size (Cohen's *d*) for the *t* test.

```
install.packages('lsr')
library(lsr)
```

Then, subset the points to test for differences between the points of the two conditions.

```
ona_first_points_d1 = as.matrix(set.ona$points$Condition$FirstGame)[,1]
ona_second_points_d1 = as.matrix(set.ona$points$Condition$SecondGame)[,1]

ona_first_points_d2 = as.matrix(set.ona$points$Condition$FirstGame)[,2]
ona_second_points_d2 = as.matrix(set.ona$points$Condition$SecondGame)[,2]
```

Conduct the *t* test on the first and second dimensions.

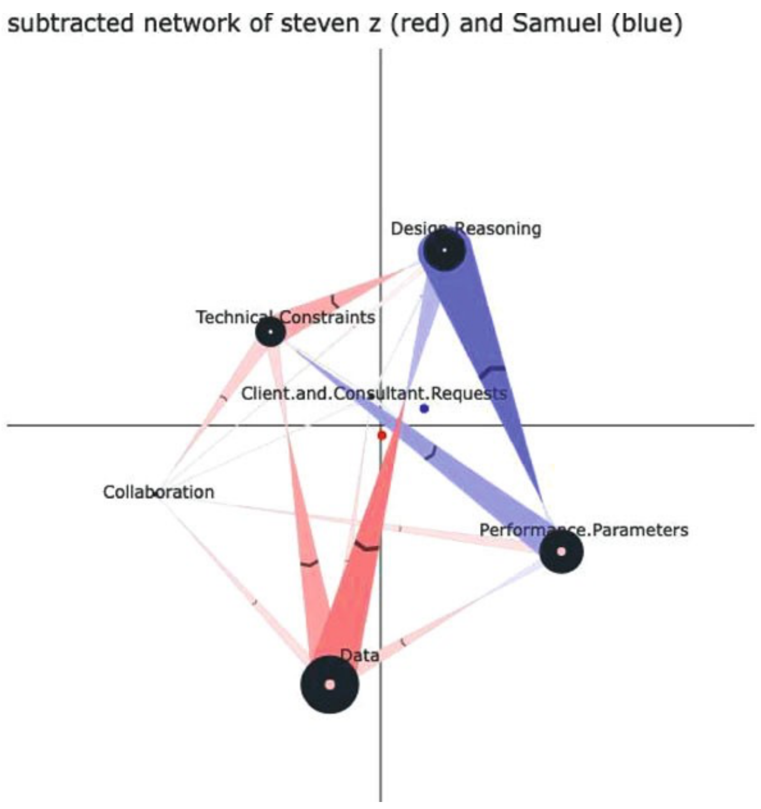

subtracted network of steven z (red) and Samuel (blue)
Design.Reasoning
Technical.Constraints
Client.and.Consultant.Requests
Collaboration
Performance.Parameters
Data

Fig. 24 ONA subtracted network showing the differences between one student from FirstGame (red) and another student from SecondGame (blue)

```
parametric tests
t_test_d1 = t.test(ona_first_points_d1, ona_second_points_d1)
t_test_d1
```

```
##
##  Welch Two Sample t-test
##
## data:  ona_first_points_d1 and ona_second_points_d1
## t = -3.7729, df = 41.001, p-value = 0.0005111
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.18227713 -0.05517572
## sample estimates:
##   mean of x   mean of y
## -0.05441628  0.06431015
```

```
t_test_d2 = t.test(ona_first_points_d2, ona_second_points_d2)
t_test_d2
```

```
##
##  Welch Two Sample t-test
##
## data:  ona_first_points_d2 and ona_second_points_d2
## t = -6.9301e-16, df = 45.45, p-value = 1
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
```

```
##  -0.1008208  0.1008208
## sample estimates:
##     mean of x     mean of y
## -1.727628e-17  1.742362e-17
```

Compute any other statistics that may be of interest. A few examples are given below.

```
mean(ona_first_points_d1)
```

```
## [1] -0.05441628
```

```
mean(ona_second_points_d1)
```

```
## [1] 0.06431015
```

```
mean(ona_first_points_d2)
```

```
## [1] -1.727628e-17
```

```
mean(ona_second_points_d2)
```

```
## [1] 1.742362e-17
```

```
sd(ona_first_points_d1)
```

```
## [1] 0.09754142
```

```
sd(ona_second_points_d1)
```

```
## [1] 0.1171941
```

```
sd(ona_first_points_d2)
```

```
## [1] 0.1784777
```

```
sd(ona_second_points_d2)
```

```
## [1] 0.1679372
```

```
length(ona_first_points_d1)
```

```
## [1] 26
```

```
length(ona_second_points_d1)
```

```
## [1] 22
```

```
length(ona_first_points_d2)
```

```
## [1] 26
```

```
length(ona_second_points_d2)
```

```
## [1] 22
```

```
cohensD(ona_first_points_d1, ona_second_points_d1)
```

```
## [1] 1.109985
```

```
cohensD(ona_first_points_d2,  ona_second_points_d2)
```

```
## [1] 1.997173e-16
```

Here, along the *x* axis (MR1), a two-sample *t* test assuming unequal variance shows that the FirstGame (mean = −0.05, SD = 0.09, N = 26) condition is statistically significantly different for alpha = 0.05 from the SecondGame condition (mean = 0.06, SD = 0.12, N = 22; t(41.001) = −3.77, p = 0.00, Cohen's d = 1.1). Along the *y* axis (SVD2), a two-sample *t* test assuming unequal variance shows that the FirstGame condition (mean = −1.73, SD = 0.17, N = 26) is not statistically significantly different for alpha = 0.05 from the SecondGame condition (mean = 1,74, SD = 0.17, N = 22; t(45.45) = 0, p = 1.00, Cohen's d = 0.00).

The Mann-Whitney *U* test is a non-parametric alternative to the independent two-sample *t* test.

First, install the rcompanion package to calculate the effect size (*r*) for a Mann-Whitney *U* test.

```
# install.packages('rcompanion')
library(rcompanion)
```

Then, conduct a Mann-Whitney *U* test on the first and second dimensions.

```
# non parametric tests
w_test_d1 = wilcox.test(ona_first_points_d1, ona_second_points_d1)
w_test_d2  = wilcox.test(ona_first_points_d2,  ona_second_points_d2)

w_test_d1
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  ona_first_points_d1 and ona_second_points_d1
## W = 130, p-value = 0.0009533
## alternative hypothesis: true location shift is not equal to 0
```

```
w_test_d2
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  ona_first_points_d2 and ona_second_points_d2
## W = 264, p-value = 0.6593
## alternative hypothesis: true location shift is not equal to 0
```

Compute any other statistics that may be of interest. A few examples are given below.

```
median(ona_first_points_d1)
```

```
## [1] -0.04307778
```

```
median(ona_second_points_d1)
```

```
## [1] 0.09596238
```

```
median(ona_first_points_d2)
```

```
## [1] 0.001753116
```

```
median(ona_second_points_d2)
```

```
## [1] 0.05862436
```

```
length(ona_first_points_d1)
```

```
## [1] 26
```

```
length(ona_second_points_d1)
```

```
## [1] 22
```

```
length(ona_first_points_d2)
```

```
## [1] 26
```

```
length(ona_second_points_d2)
```

```
## [1] 22
```

```
abs(wilcoxonR(ona_first_points_d1,  ona_second_points_d1))
```

```
## r
## 0
```

```
abs(wilcoxonR(ona_first_points_d2,  ona_second_points_d2))
```

```
##     r
## 0.707
```

Here, along the *x* axis (MR1), a Mann-Whitney *U* test shows that the FirstGame condition (Mdn = −0.04, N = 26) was statistically significantly different for alpha = 0.05 from the SecondGame condition (Mdn = 0.10, N = 22 U = 130, p = 0.001, r = 0.00). Along the *y* axis (SVD2), a Mann-Whitney *U* test shows that the FirstGame condition (Mdn = 0.001, N = 26) is not statistically significantly different for alpha = 0.05 from the SecondGame condition (Mdn = 0.00, N = 22, U = 264, p = 0.66, r = 0.71). The absolute value of r value in Mann-Whitney U test varies from 0 to close to 1. The interpretation values for r commonly in published literature is: 0.10 - < 0.3 (small effect), 0.30 - < 0.5 (moderate effect) and > = 0.5 (large effect).

4.7 Model Evaluation

4.7.1 Variance Explained

Briefly, variance explained (also called explained variation) refers to the proportion of the total variance in a dataset that is accounted for by a statistical model or set of predictors.

In ONA, to represent high-dimensional vectors in a two-dimensional space, ONA uses either singular value decomposition or means rotation combined with SVD. For each of the reduced dimensions, the variance in patterns of association among units explained by that dimension can be computed.

```
head(set.ona$model$variance,2)
```

```
##       MR1      SVD2
## 0.1367940 0.2736079
```

In our example above, since we used means rotation method, the first dimension is labeled as MR1 and the second dimension is labeled as SVD2.The two dimensions in combination explained more than 40% of the variance.

Here, the first dimension is MR1 and the second dimension is SVD2. The two dimensions in combination explained more than 40% of the variance.

As with any statistical model, greater explained variance does not necessarily indicate a better model, as it may be due to overfitting, but it provides one indicator of model quality.

4.7.2 Goodness of Fit

Briefly, a model's goodness of fit refers to how well a model fits or represents the data. A model with a high goodness of fit indicates that it accurately represents the data and can make reliable predictions.

In ONA, a good fit means that the positions of the nodes in the space—and thus the network visualizations—are consistent with the mathematical properties of the model. In other words, we can confidently rely on the network visualizations to interpret the ONA model. The process that ONA uses to achieve high goodness of fit is called co-registration, the same as the one used in ENA. The mathematical details of co-registration are beyond the scope of this chapter and can be found in Bowman et al. [2].

To test a model's goodness of fit, use ona::correlations. The closer the value is to 1, the higher the model's goodness of fit is. Most ENA models have a goodness of fit that is well above 0.90.

```
ona::correlations(set.ona)
```

```
##     pearson  spearman
## 1 0.9801173 0.9801799
## 2 0.9801431 0.9759160
```

4.7.3 Close the Interpretative Loop

Another approach to evaluate an ONA model is to confirm the alignment between quantitative model (in our case, our ONA model) and the original qualitative data. In other words, we can return to the original data to confirm that quantitative findings give a fair representation of the data. This approach is an example of what's called as closing the interpretative loop in Quantitative Ethnography field [1].

For example, based on our visual analysis of the network of "SecondGame::samuel o" in previous section, we are interested in what the lines are in the original data that contributed to the connection *from* Performance.Parameters *to* Design.Reasoning.

Let's first review what "SecondGame::samuel o" ONA network looks like. Based on the connection direction and strength from Technical.Constraints to Performance.Parameters, we would expect to see more examples of Samuel responded with "Design.Reasoning" to "Performance.Parameters", than the other way around (Fig. 25).

```
ona:::plot.ena.ordered.set(set.ona, title = "SecondGame::samuel o") |>
  units(
    points=set.ona$points$ENA_UNIT$`SecondGame::samuel o`,
    points_color = "blue",
    show_mean = FALSE, show_points = TRUE, with_ci = FALSE) |>
  edges(
    weights = set.ona$line.weights$ENA_UNIT$`SecondGame::samuel o`,
    edge_size_multiplier = edge_size_multiplier,
    edge_arrow_saturation_multiplier = edge_arrow_saturation_multiplier,
    node_position_multiplier = node_position_multiplier,
    edge_color = c("blue")) |>
  nodes(
    node_size_multiplier = node_size_multiplier,
    node_position_multiplier = node_position_multiplier,
    self_connection_color = c("blue"))
```

To do so, we use view() function and specify required parameters as below.

This is going to activate a window shows up in your Viewer panel. If it is too small to read, you can click on the "Show in new window" button to view it in your browser for better readability.

In the Viewer panel, hover over your cursor on any of the lines that are in bold, a size of 7 lines rectangle shows up, representing that in a moving stanza window of size 7, the referent line (the line in bold) and its preceding 6 lines. The 1 and 0 in Technical.Constraints column and Design.Reasoning column shows where the connections happened (Fig. 26).

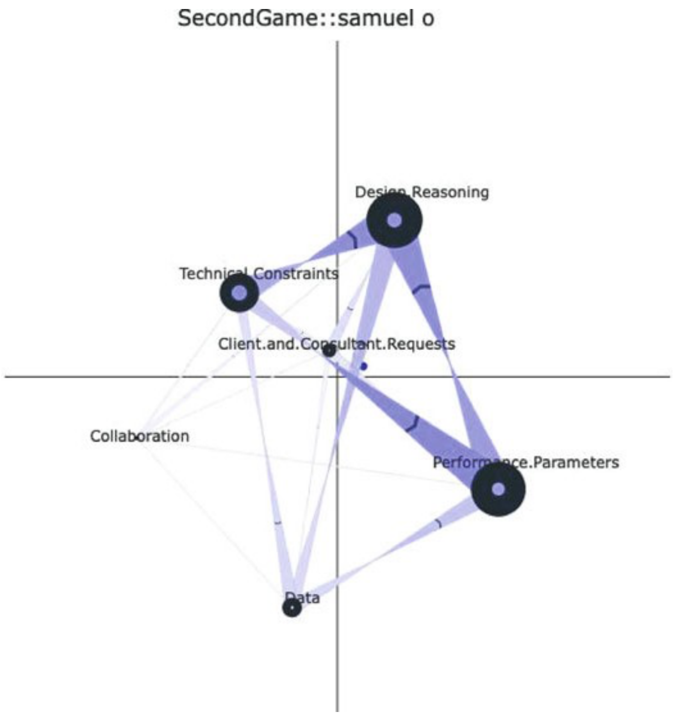
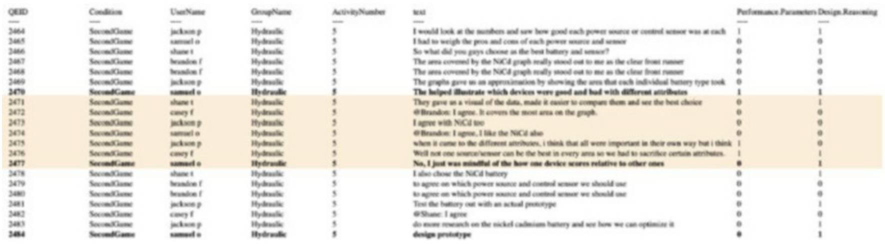


Fig. 25 ONA network for a student from SecondGame



QEID	Condition	UserName	GroupName	ActivityNumber	text	Performance.Parameters	Design.Reasoning
2464	SecondGame	jackson p	Hydraulic	5	I would look at the numbers and saw how good each power source or control sensor was at each	1	1
2465	SecondGame	samuel o	Hydraulic	5	I had to weigh the pros and cons of each power source and sensor	0	0
2466	SecondGame	shane t	Hydraulic	5	So what did you guys choose as the best battery and sensor?	0	1
2467	SecondGame	brandon f	Hydraulic	5	The area covered by the NiCd graph really stood out to me as the clear front runner	0	0
2468	SecondGame	brandon f	Hydraulic	5	The area covered by the NiCd graph really stood out to me as the clear front runner	0	0
2469	SecondGame	jackson p	Hydraulic	5	The graphs gave us an approximation by showing the area that each individual battery type took	0	0
2470	**SecondGame**	**samuel o**	**Hydraulic**	**5**	**The helped illustrate which devices were good and bad with different attributes**	**1**	**1**
2471	SecondGame	shane t	Hydraulic	5	They gave us a visual of the data, made it easier to compare them and see the best choice	0	1
2472	SecondGame	casey f	Hydraulic	5	@Brandon: I agree. It covers the most area on the graph.	0	0
2473	SecondGame	jackson p	Hydraulic	5	I agree with NiCd too	0	0
2474	SecondGame	samuel o	Hydraulic	5	@Brandon: I agree, I like the NiCd also	0	0
2475	SecondGame	jackson p	Hydraulic	5	when it came to the different attributes, i think that all were important in their own way but i think	1	0
2476	SecondGame	casey f	Hydraulic	5	Well not one source/sensor can be the best in every area so we had to sacrifice certain attributes.	1	1
2477	**SecondGame**	**samuel o**	**Hydraulic**	**5**	**No, I just was mindful of the how one device scores relative to other ones**	**0**	**1**
2478	SecondGame	shane t	Hydraulic	5	I also chose the NiCd battery	0	1
2479	SecondGame	brandon f	Hydraulic	5	to agree on which power source and control sensor we should use	0	0
2480	SecondGame	brandon f	Hydraulic	5	to agree on which power source and control sensor we should use	0	0
2481	SecondGame	jackson p	Hydraulic	5	Test the battery out with an actual prototype	0	1
2482	SecondGame	casey f	Hydraulic	5	@Shane: I agree	0	0
2483	SecondGame	jackson p	Hydraulic	5	do more research on the nickel cadmium battery and see how we can optimize it	0	1
2484	**SecondGame**	**samuel o**	**Hydraulic**	**5**	**design prototype**	**0**	**1**

Fig. 26 A screenshot of the view() function result. The highlighted lines represent lines within the same stanza window

Notice that here we are viewing the same qualitative example as in Sect. 3.7.3 in ENA. In line 2477 Samuel shared his [Design.Reasoning] about "mindful of (the) how one device scores relative to other ones", ***as a response to*** what Casey said in line 2476 about [Performance.Parameters] "not one source/censor can be the best in every area so we had to sacrifice certain attributes", as well as what Jackson said in line 2475 about safety as one of the [Performance.Parameters] "when it came to the different attributes, i think that all were important in their own way but i think safety is one of the most important".

Here, ONA was able to not only capture the occurrence between code Design.Reasoning and Performance.Parameters as ENA did, but also represent the connection direction *from* Design.Reasoning *to* Performance.Parameters.

4.8 Using ONA Model Outputs in Other Analyses

As with ENA, the outputs of ONA models can be used as inputs in other statistical models. See Sect. 3.8 for an example using ENA points.

5 Additional Features

In the sections above, we demonstrated how to do an ENA analysis and an ONA analysis. In this section, we show how to project new data into a space constructed with different data. This can be done as long as the same codes are used in both sets.

5.1 Projections in ENA

To project the ENA points from one model into a space constructed with different data, replace the rotation.set parameter of ena.make.set. In the example below, an "expert" model is developed using the SecondGame units and the FirstGame (novice) units are projected into that space. By projecting novice model's units into expert model's space, users can interpret the projected novice units' networks based on the two dimensions defined by expert model's node positions. In other words, interpreting novice's networks in the context of experts' space (Figs. 27 and 28).

```r
data = rENA::RS.data

#expert data
exp.data = subset(data, Condition == "SecondGame")

#novice data
nov.data = subset(data, Condition == "FirstGame")

#expert model
units_exp = exp.data[,c("Condition","UserName")]
conversation_exp = exp.data[,c("Condition","GroupName","ActivityNumber")]
codes_exp = exp.data[,codeCols]
meta_exp = exp.data[,c("CONFIDENCE.Change",
                  "CONFIDENCE.Pre","CONFIDENCE.Post","C.Change")]

set_exp =
  ena.accumulate.data(
  text_data = exp.data[, 'text'],
  units = units_exp,
  conversation = conversation_exp,
  codes = codes_exp,
  metadata = meta_exp,
  window.size.back = 7,
) |>
  ena.make.set()

#novice model
units_nov = nov.data[,c("Condition","UserName")]
conversation_nov = nov.data[,c("Condition","GroupName","ActivityNumber")]
codes_nov = nov.data[,codeCols]
meta_nov = nov.data[,c("CONFIDENCE.Change",
                  "CONFIDENCE.Pre","CONFIDENCE.Post","C.Change")]

set_nov =
  ena.accumulate.data(
  text_data = nov.data[, 'text'],
  units = units_nov,
  conversation = conversation_nov,
  codes = codes_nov,
  metadata = meta_nov,
  window.size.back = 7,
) |>
  ena.make.set(rotation.set = set_exp$rotation)


# plot expert model (what we projected into) Using plotting wrapper to save t
ime
plot_exp = ena.plotter(set_exp,
                        points = T,
                         mean = T,
                         network = T,
                         print.plots = F
                        )

# plot test model (points from test model in training model space)
plot_nov = ena.plotter(set_nov,
                         points = T,
                         mean = T,
                         network = T,
                         print.plots = F)

#compare plots
plot_exp$plot
```

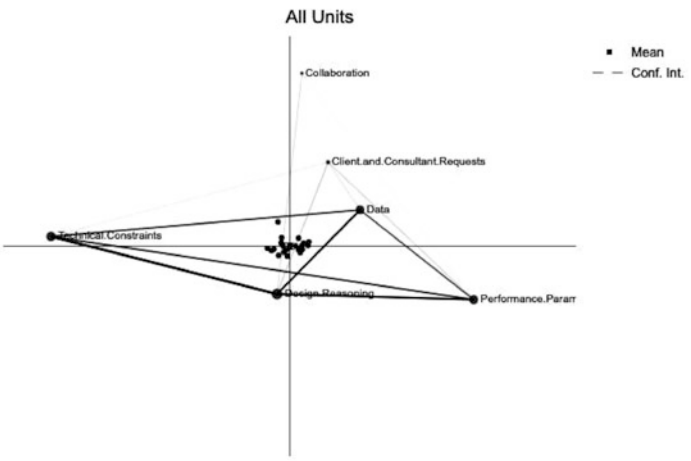


Fig. 27 Mean network for all units in the expert model

```
plot_nov$plot
```

5.2 *Projections in ONA*

Projection works similarly in ONA (Fig. 29).

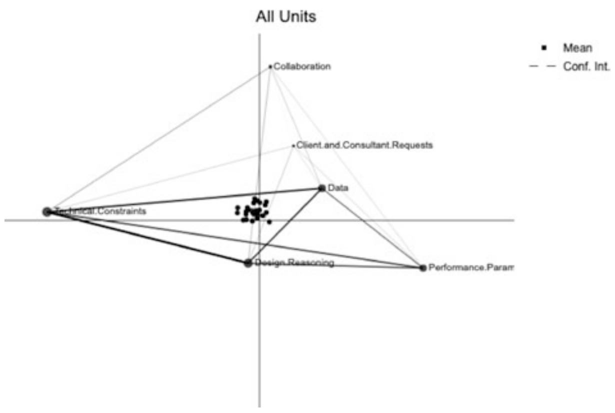


Fig. 28 Mean network for novice students projected into expert space

```
data = ona::RS.data

#expert data
exp.data = subset(data, Condition == "SecondGame")

#novice data
nov.data = subset(data, Condition == "FirstGame")

#shared unit cols
units = c("UserName","Condition","GroupName")

#shared code cols
codes = c(
          'Data',
          'Technical.Constraints',
          'Performance.Parameters',
          'Client.and.Consultant.Requests',
          'Design.Reasoning',
          'Collaboration')
```

```
#shared hoo
hoo = conversation_rules(
  (Condition %in% UNIT$Condition & GroupName %in% UNIT$GroupName))


#expert accum
accum.exp = contexts(exp.data, units_by = units, hoo_rules = hoo) |>
  accumulate_contexts(codes = codes,
                      decay.function = decay(simple_window, window_size = 7),
                      return.ena.set = FALSE, norm.by = NULL)
#expert model
set.exp = model(accum.exp)

#novice accum
accum.nov = contexts(nov.data, units_by = units, hoo_rules = hoo) |>
  accumulate_contexts(codes = codes,
                      decay.function = decay(simple_window, window_size = 7),
                      return.ena.set = FALSE, norm.by = NULL)
#novice model
set.nov = model(accum.nov)

# projecting novice data into expert space
set = model(accum.nov, rotation.set = set.exp$rotation)

ona:::plot.ena.ordered.set(set, title = "novice data into expert space") |>
  units(
    points = set$points,
    show_mean = TRUE, show_points = TRUE, with_ci = TRUE) |>
  edges(
    weights = set$line.weights) |>
  nodes(
    self_connection_color = "red",
    node_size_multiplier = 0.6)
```

6 Discussion

In this chapter, we introduced two techniques, ENA and ONA, for quantifying, visualizing, and interpreting networks using coded data. Through the use of a demonstration dataset that documents collaborative discourse among students collaborating to solve an engineering design problem, we provided step-by-step instructions on how to model complex, collaborative thinking using ENA and ONA in R. The chapter combines theoretical explanations with tutorials, intended to be of aid to researchers with varying degrees of familiarity with network analysis techniques and R. This chapter mainly showcased the standard and most common use of these two tools. The ENA and ONA R packages, akin to other R packages, offer flexibility to researchers to tailor their analyses to their specific needs. For example, users with advanced R knowledge can supply their own adjacency matrices and use ENA or ONA solely as a visualization tool rather than an integrated modeling and visualization tool.

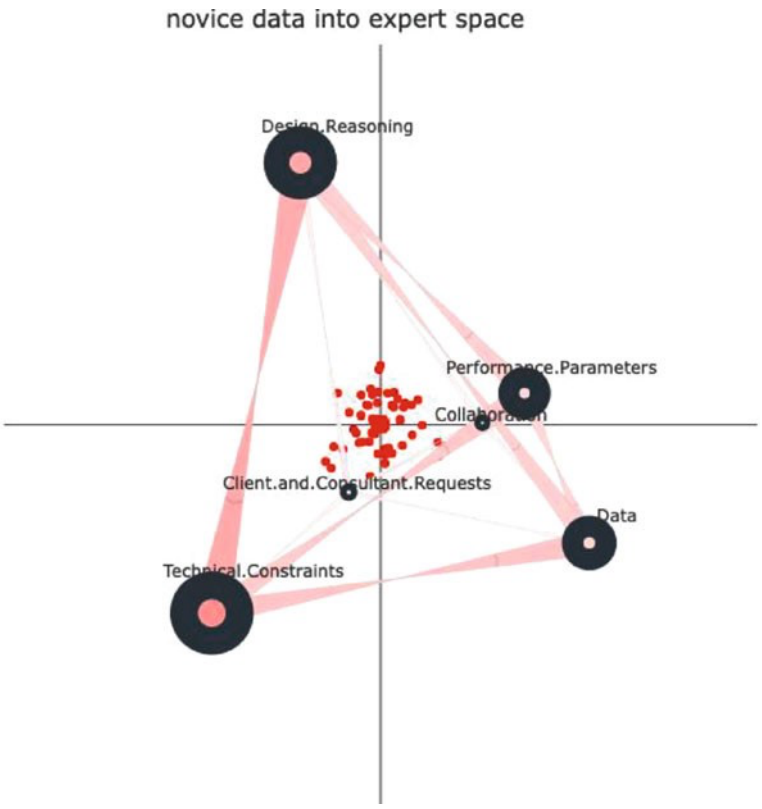


Fig. 29 Projecting novice data into expert space

Due to the technical and practical focus of this chapter, we omitted detailed explanations of the theoretical, methodological, and mathematical foundations of ENA and ONA that are crucial for informed, theory-based learning analytics research using these techniques. Consult the Further Reading section for papers that explain these aspects of ENA and ONA in greater detail.

References

1. Shaffer DW (2017) Quantitative ethnography. Cathcart Press
2. Bowman D, Swiecki Z, Cai Z, Wang Y, Eagan B, Linderoth J, Shaffer DW (2021) The mathematical foundations of epistemic network analysis. In: Advances in Quantitative Ethnography: Second International Conference, ICQE 2020, Malibu, CA, USA, February 1–3, 2021, Proceedings 2, pp 91–105
3. Csanadi A, Eagan B, Shaffer DW, Kollar I, Fischer F (2018) When coding-and-counting is not enough: using epistemic network analysis (ENA) to analyze verbal data in CSCL research. Int J Comput-Support Collab Learn 13(4):419–438

4. Oshima J, Oshima R, Fujita W (2018) A mixed-methods approach to analyze shared epistemic agency in Jigsaw instruction at multiple scales of temporality. J Learn Anal *5*(1):10–24. https://doi.org/10.18608/jla.2018.51.2
5. Bressler DM, Bodzin AM, Eagan B, Tabatabai S (2019) Using epistemic network analysis to examine discourse and scientific practice during a collaborative game. J Sci Educ Technol *28*:553–566
6. Swiecki Z, Ruis AR, Farrell C, Shaffer DW (2020) Assessing individual contributions to collaborative problem solving: a network analysis approach. Comput Hum Behav 104:105876
7. Prieto LP, Rodríguez-Triana MJ, Ley T, Eagan B (2021) The value of epistemic network analysis in single-case learning analytics: a case study in lifelong learning. In: Advances in quantitative ethnography: second international conference, ICQE 2020, Malibu, CA, USA, February 1–3, 2021, Proceedings 2, Springer International Publishing, pp. 202–217
8. Zhang S, Gao Q, Sun M, Cai Z, Li H, Tang Y, Liu Q (2022) Understanding student teachers' collaborative problem solving: Insights from an epistemic network analysis (ENA). Comput Educ 183:104485
9. Bauer E, Sailer M, Kiesewetter J, Shaffer DW, Schulz C, Pfeiffer J, Gurevych I, Fischer MR, Fischer F (2020) Pre-Service teachers' diagnostic argumentation: what is the role of conceptual knowledge and epistemic activities?. In: Proceedings of the fifteenth international conference of the learning sciences, pp 2399–2400.
10. Fernandez-Nieto GM, Martinez-Maldonado R, Kitto K, Buckingham Shum S (2021) Modelling spatial behaviours in clinical team simulations using epistemic network analysis: methodology and teacher evaluation. In: LAK21: 11th international learning analytics and knowledge conference, April, pp 386–396
11. Phillips M, Trevisan O, Carli M, Mannix T, Gargiso R, Gabelli L, Lippiello S (2023, March) Uncovering patterns and (dis) similarities of pre-service teachers through Epistemic Network Analysis. In: Society for information technology & teacher education international conference. Association for the Advancement of Computing in Education (AACE), pp 1021–1030
12. Tan Y, Ruis AR, Marquart C, Cai Z, Knowles MA, Shaffer DW (2022, October) Ordered network analysis. In: International conference on quantitative ethnography. Springer Nature Switzerland, Cham, pp 101–116
13. Fan Y, Tan Y, Rakovic´ M, Wang Y, Cai Z, Shaffer DW, Gaševic´ D (2022) Dissecting learning tactics in MOOC using ordered network analysis. J Comput Assist Learn:1–13
14. Shaffer DW, Collier W, Ruis AR (2016) A tutorial on epistemic network analysis: analyzing the structure of connections in cognitive, social, and interaction data. J Learn Anal 3(3):9–45
15. Shaffer DW, Arastoopour G (2014) Guide to RSdata.csv sample ENA data set. Madison, WI: Games and Professional Simulations Technical Report 2014-3
16. Arastoopour G, Shaffer DW, Swiecki Z, Ruis AR, Chesler NC (2016) Teaching and assessing engineering design thinking with virtual internships and epistemic network analysis. Int J Eng Educ 32(3B):1492–1501
17. Chesler NC, Ruis AR, Collier W, Swiecki Z, Arastoopour G, Shaffer DW (2015) A novel paradigm for engineering education: Virtual internships with individualized mentoring and assessment of engineering thinking. J Biomech Eng *137*(2)
18. Siebert-Evenstone AL, Arastoopour Irgens G, Collier W, Swiecki Z, Ruis AR, Shaffer DW (2017) In search of conversational grain size: Modelling semantic structure using moving stanza windows. J Learn Anal 4(3):123–139
19. Ruis AR, Siebert-Evenstone AL, Pozen R, Eagan B, Shaffer DW (2019) Finding common ground: a method for measuring recent temporal context in analyses of complex, collaborative thinking. In: Lund K, Niccolai G, Lavoué E, Hmelo-Silver C, Gwon G, Baker M (eds) A wide lens: combining embodied, enactive, extended, and embedded learning in collaborative settings: 13th international conference on Computer Supported Collaborative Learning (CSCL), I, pp 136–143
20. Kaur A, Kumar R (2015) Comparative analysis of parametric and non-parametric tests. J Comput Math Sci *6*(6):336–342

Further Reading

21. Brohinsky J, Marquart C, Wang J, Ruis AR, Shaffer DW (2021) Trajectories in epistemic network analysis. In Ruis AR, Lee SB (eds) Advances in quantitative ethnography: second international conference, ICQE 2020, Malibu, CA, USA, February 1–3, 2021, Proceedings, Springer, pp 106–121
22. Gasevic D, Greiff S, Shaffer DW (2022) Towards strengthening links between learning analytics and assessment: challenges and potentials of a promising new bond. Comput Hum Behav:1–7
23. Shaffer DW (2017) Quantitative ethnography. Cathcart Press
24. Shaffer DW (2018) Big data for thick description of deep learning. In: Millis K, Long D, Magliano J, Weimer K (eds) Deep learning: multi-disciplinary approaches. Routledge, New York, pp 262–275
25. Shaffer DW (2018) Epistemic network analysis: understanding learning by using big data for thick description. In: Fischer F, Hmelo-Silver CE, Goldman SR, Reimann P (eds) International handbook of the learning sciences. Routledge, New York, pp 520–531
26. Shaffer DW, Eagan B, Knowles M, Porter C, Cai Z (2022) Zero re-centered projection: an alternative proposal for modeling empty networks in ENA. In: Wasson B, Zörgo″ S (eds) Advances in quantitative ethnography: third international conference, ICQE 2021, Virtual Event, November 6–11, 2021, Proceedings, Springer, pp 66–79
27. Swiecki Z, Lian Z, Ruis AR, Shaffer DW (2019) Does order matter? Investigating sequential and cotemporal models of collaboration. In: Lund K, Niccolai G, Lavoué E, Hmelo-Silver C, Gwon G, Baker M (eds) A wide lens: combining embodied, enactive, extended, and embedded learning in collaborative settings: 13th international conference on Computer Supported Collaborative Learning (CSCL), I, pp 112–119
28. Wang Y, Swiecki Z, Ruis AR, Shaffer DW (2021) Simplification of epistemic networks using parsimonious removal with interpretive alignment. In Ruis AR, Lee SB (eds) Advances in quantitative ethnography: second international conference, ICQE 2020, Malibu, CA, USA, February 1–3, 2021, Proceedings, Springer, pp 137–151