hENA, and other Cans

Mariah A. Knowles
Library and Information Science
University of Wisconsin–Madison
she/her/hers

David W. Shaffer
Educational Psychology
University of Wisconsin–Madison
he/him/his

January 10, 2021

Contents

1 Introduction **2**

2 High-Dimensional Spaces **3**
2.1 C&C Biplot 3
2.2 Epistemic Networks 5

3 Empty Units **7**

4 Geometric Metaphor **8**

5 Rotations **11**
5.1 Top-Level Rotation Types 11
5.2 Regression Framework 12
5.3 Family Tree 12
5.4 Controls etc. 14

6 Visualization **14**
6.1 Attempts so Far 14
6.2 Remaining Issues 17

7 jENA API **18**
7.1 Minimal Examples 18
7.2 Options 20
7.3 Infrastructure 21
7.4 Model Computation 23

8 CUSUM 24

1 Introduction

Between May and December 2020, I was part of an 80 hour research practicum with the Epistemic Analytics lab.

My initial goal in the lab was to understand how to do ENA rotations while controlling for arbitrary variables. David named this extension "hENA," for hierarchical epistemic network analysis. I was interested in sociolinguistic variables, such as age or race or gender or SES, which might correlate with (perceptions of) speaking patterns (either from peers or from the researchers), while also correlating with the grouping variable (those that scored high on a test vs. those that scored low). David was interested in controlling for the effect of nested data, which is common in educational settings, eg, students in groups in classes in schools in districts.

In plain ENA at the time, we could do a rotation that shows the difference between two groups, namely a means rotation (MR1). And we could test that those groups, on average, had different x positions, even when controls were included—such as controls for the effect of students being at the same school. The problem with this, I believed, was that the ENA plot showed one thing, and the test was of a different thing. This undermined the benefit of ENA in showing that there is a pattern and what that pattern was.

I wrote my first attempt at implementing a controlled rotation—I called it a moderated means rotation (MMR1)—during my QE class with David the spring before the practicum. That solution was written in R and was based directly on the open source rENA package distributed through CRAN.

Feeling limited in R and wanting to have greater freedom in my approach to ENA, I wrote my own implementation of ENA in Julia. I wrote this implementation from scratch twice: the first time based on David's descriptions during our Teams meetings of how ENA worked and checking my results against the equivalent API calls in rENA; and again when I realized that the Julia implementation would be faster and more interoperable with existing Julia libraries if I centered it around a composed type hierarchy.

As I worked towards implementing controlled rotations and developing clear statistical, geometric, and ethnographic interpretations of the approach, we opened a few other cans, many of which are still open and unsolved.

So, the purpose of this report is to explain (i) the statistical, geometric, and ethnographic facets of my approach to hENA; (ii) the structure of my implementation of ENA in Julia; and (iii) the remaining questions and our attempts and failures at answering each of them.

2 High-Dimensional Spaces

ENA is one method for modeling connections between qualitative codes, and it produces an interpretable plot and scores that can be used to test hypotheses. ENA is not the only method that can produce these things; one might decide to use a simpler, "code and count" model and plot it using a standard or rotated biplot. There would be obvious limitations that what the biplot models would be "thinner" and based on the assumption that the codes matter even in isolation from one another.

I believe that the generalization between these two is still instructive, since what matters is, in either approach, we produce a high dimensional space, we use unit positions within that space to test that there is a pattern in the data, we use the positions of units and variables in the space to show what that pattern is, and we are given a method by which to compare an arbitrary number of units at once.

2.1 C&C Biplot

Let H be a matrix representation of a high dimensional space, such that the ith row corresponds to the ith unit and the jth row corresponds to the jth qualitative code. That is, H_{ij} is proportional to the number of cases where unit i make connections with code j. We call this a "code and count" model (C&C).

H can be normalized or unnormalized.

When unnormalized, the vector H_i will be further from zero for units who talk more, for whom we have more data, etc., compared to those who talk less, etc.

When normalized, all vectors H_i have been made unit-length. Thus, if Betty and Veronica make connections with the same codes in the same proportions, then they will lie at the same position in the high dimensional space, even if Betty talks more. In this case, vectors are sphere normed such that:

$$\forall_i H_i := \frac{H_i}{\sqrt{\sum_j H_{ij}^2}} \tag{1}$$

Regardless of norming, units and codes can be plotted as a standard or rotated biplot. Figure 1 gives an example rotated biplot that demonstrates *that* there is a pattern in the data that corresponds to time, and *what* that pattern is: Over the course of this semester, my skin has gotten clearer (OILY is to the left of the plot) and I've been less anxious (NONHAPPY is left-of-center and HAPPY is far to the right).

A limitation of the C&C biplot approach is that what it models is *thin*. There are no connections between codes. It is able to test simple claims about my mood and my skin since this summer, but it lacks any richness or insight. Counting codes in isolation is hardly adequate. A better model would, among other things, capture the *thick* qualitative connections between my prescriptions, my physical changes, my anxious immediate thoughts, and my interactions with others. A

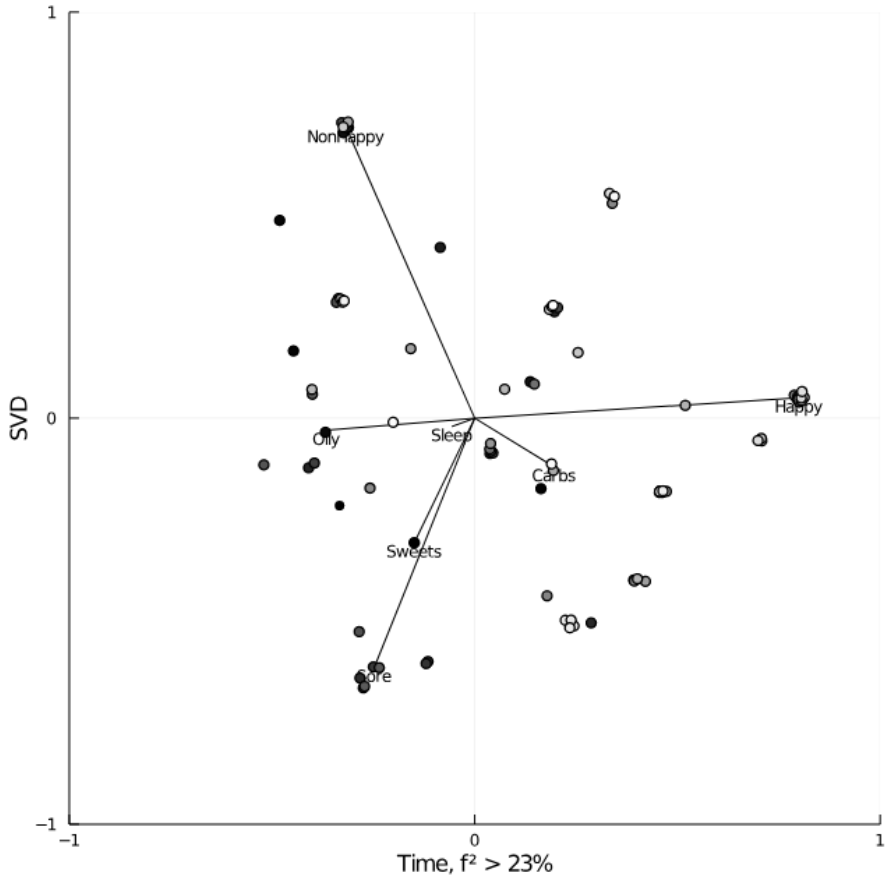

Figure 1: Example C&C Biplot. White circles with black outlines represent units (days). Units are positioned on the plot with a small random jitter. Lines with black dots represent variables (codes). The biplot is rotated such that units/variables further to the left, on average, correspond with earlier dates, and vice versa. The y-axis is chosen to be the highest variance dimension in the space that is also orthogonal to the x-axis; this just gives a more legible plot. An f^2 test shows a medium effect size between time and the account of the unit's codes given by the x-axis.

C&C biplot approach *can* achieve this. But it would be a big lift, even if it is a useful first model to start with.

2.2 Epistemic Networks

Epistemic Network Analysis (ENA) takes the C&C and biplot approached one step further. Instead of using the original codes as the variables for H, it uses the *connections* between those codes.

That is, H_{ij} does not model the ith unit's connections to the jth code, but instead to the jth *pair* of codes.

If there are N codes, then there are up to $(N^2 - N)/2$ pairs of codes in an undirected ENA model. Others in this lab have done work on directed models that would give up to N^2 pairs instead.

Regardless of directedness, ENA also produces a second high-dimensional space, $\hat{H}$, an approximation of H.

H contains the units and the code-pairs, but $\hat{H}$ also contains the original N codes. This is necessary in order to label and interpret H in the original qualitative terms. Within H, units are placed at their weighted sum of the code-pair positions within the space. Within $\hat{H}$, units are placed at the "center of mass" of their networks of connections between the original N codes. Figure 2 gives an example. ENA plots are similar to ordinary biplots in that it simultaneously shows units and variables. In an ENA plot, the midpoint of any pair of codes is *approximate* to where that code-pair would be placed were we to plot ENA's H using a standard biplot. ENA plots, however, allow us to shift our interpretation to the original N codes and the networks of connections between them; it was these connections that the C&C biplot was lacking at the end of the previous section.

Generally, ENA produces an $\hat{H}$ with a high coregistration with H along the axes of interest; however, we have seen lower coregistrations when using a directed model, and this is an ongoing conversation.

Others in the lab find $\hat{H}$ after rotating, but the order does not matter, and I prefer to find $\hat{H}$ first because it allows me to define a rotation more arbitrarily, ie, in terms of units in H, units in $\hat{H}$, or codes in $\hat{H}$.

To compute the approximation $\hat{H}$, one needs to find a best fit position of each code along each column of H. To do this, one first needs to find the *networked weight* of each code for each unit; this is different from the *weight in isolation* of the code used in the C&C model. Instances of a code that make no connections to other codes in the network, even if appearing often in isolation, cannot contribute to that code's network weight.

Let C_{ij} represent these network weights:

$$\forall_{ij} C_{ij} = \sum_{k \texttt{ connecting to } j} \frac{H_{ik}}{2} \tag{2}$$

Then regress each column of H by the no-intercept[1] regression:

[1] I have worked out why we need to use the no-intercept model on paper, based on a description of the method by David in a Teams call. Since others in our lab have shown this, I won't here.

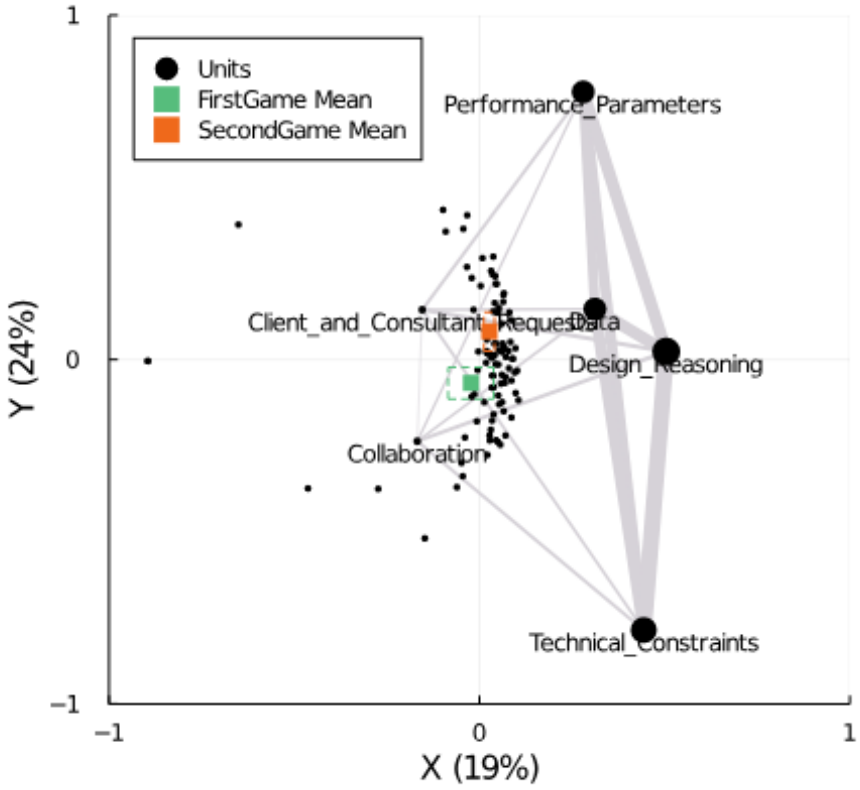

Figure 2: Example ENA plot. Small black dots represent units (students). Large black dots represent original N codes, and the grey lines between them represent the relative weight of their interconnections, on average across all units. The green and orange squares represent the mean positions of two groups of units, and the dashed box around them represent the confidence intervals of those means. The space has been rotated so that the x-axis models the zero-to-non-zero-mean direction and the y-axis captures the highest variance orthogonal to the x-axis. Much has been written about this dataset elsewhere.

$$\forall_j H_j^T \sim 0 + C_1^T + C_2^T + \ldots + C_N^T \tag{3}$$

The position of the kth code within the jth dimension of $\hat{H}$ will be given by β_k from the jth regression. That is, we can construct a matrix representation of the codes' high dimensional position. Let this be denoted by B, and let B_{kj} be given by β_k from the jth regression above. Generally though, my intuition of it is that B is embedded within $\hat{H}$.

Once B has been found, the units' positions within $\hat{H}$ can be found by finding

the "centroid" of that unit's network as positioned by the codes in B. That is, just as we currently find the centroid positions along the two plotted dimensions based on the code positions along those axes, we can find the centroid positions along each of the original dimensions of $\hat{H}$ based on the code positions along those original dimensions.

3 Empty Units

In either approach, when unit positions are normed, the empty networks become discontinuous with the interpretation of H.

In equation 1, it is possible for some H_i to be undefined because the denominator can go to zero when a unit has no codes/connections as represented in the model.

We have entertained a few solutions. This is an ongoing conversation.

The first entertained solution is to leave the empty units at zero. However, when a plot is mean-centered, then empty networks are placed unintuitively away from the origin. This may only be an issue in practice when using SVD-based rotations when there are lots of empty networks. In that case, the offending dimension is usually the first SVD dimension, and so one can discard it and plot along SVD dimensions 2 and 3 instead.

The second is to discard the empty units from the model. If permissible within one's research context, this is a simple solution.

The third is to place empty units at the "even" position, ie, at the same place as other units which have an even distribution of codes/connections. A related option is to place empty units at the "non-zero mean" position. Carl has shown that the "even" position produces worse correlations than the "non-zero mean" position.

And the fourth is to deflate H (and $\hat{H}$) so as to remove the axis that runs from zero to the mean position of the non-empty units. Figure 2 illustrates this offending axis by placing it directly along the x-axis.

One way to think of this axis is as the handle to an umbrella: when the non-empty units are normed, they are projected to a quarter-sphere in the "first quadrant" of the high dimensional space, which resembles the rain guard of an umbrella. The empty units remain at zero, which resembles the bottom end of the umbrella's handle.

If we imagine the handle running from the zero position to the non-empty mean position, and we imagine deflating that axis from the high-dimensional spaces, what we are left with is a space where zero, the origin, and the mean all align. It is like looking down at the umbrella from the top. From there we can rotate the remaining high-dimensional space (spin the high-dimensional umbrella) to highlight features of interest, just as we would in any rotation.

To deflate the space in this way, first find the direction of the umbrella handle axis in the space:

$$M = \left[E\left[H_j^T\right]\right]_j \tag{4}$$

Then normalize M to a unit-length vector and reject the direction of each of the original dimensions from M, producing a deflated space $\ddot{H}$:

$$\forall_j \ddot{H}_j^T = H_j^T - \frac{V_j \cdot M}{M \cdot M} H_j^T M \tag{5}$$

where

$$\forall_j V_j = [1 \texttt{ if } j = k \texttt{ else } 0]_k \tag{6}$$

In this deflated space $\ddot{H}$, the discontinuous distinction between zero and the non-zero mean goes away. One can also similarly deflate the space such that the zeros and origins align with, eg, the "even" unit position instead of the mean position.

However, empirically we have seen similar results between the deflated approach and the results of just dropping empty units from the model; the deflation process itself discards information about the units that we still might care about; and the deflation process is complex and difficult to explain. So, we have been inclined *not* to adopt this solution.

4 Geometric Metaphor

Imagine you are studying how students discuss their profession from the start to the end of some course. Then imagine writing your fieldnotes on sticky notes. Next, you arrange them on a markerboard such that fieldnotes from the first few meetings go on the left, and fieldnotes from the last few meetings go on the right. Finally, you summarize the patterns you see in these two groups, jotting salient codes and connections along the left and right sides of your stickynote "axis," with maybe some common patterns jotted towards the middle.

If you are correct in your interpretation of these notes, if you are correct that it is meaningful to arrange these notes this way, and if you have a grip on the data, then there is an underlying mathematical relationship.

It is this "geometric metaphor" that, for me, unites the statistical, geometric, and ethnographic interpretations of custom rotations in ENA and C&C biplot methods.

I have seen geometric thinking this way in many qualitative research papers in the *Design Studies* journal, so I'll give two examples from there.

First, in Figure 3, we see the authors producing a visual explanation of their conceptual model. This model was produced from an inductive qualitative study, and the visual was produced based on their own intuitive explanation of the events. The variables of their model are the directed connections between codes, interpreted as a causal network, and grounded in instances in their data. In other words, Van Kujik produced a model very similar to a directed ENA. Imagine the high dimensional representation of their model (H, where H_{ij} is relative to the ith instance of the jth ordered-pair of codes). The visual is a projection of that space down to a 1-dimensional plot. This projection places groups of connections into "buckets," then arranges those buckets along a single axis that represents a time progression. This axis is intended to convey the

idea that some set of events occur before some other set of events, and before another, and before another. That is, this axis is an *ordinal* space, since it conveys order and nothing else. However, I count 33 original codes in their plot, which gives over 1000 possible connections between them. And their x-axis is broken into six groups. The cardinality of their data would have to be much larger than the study they did in order to give statistical evidence that those six groups sufficiently differ. Van Kujik might be right in their explanation of their observations, and their explanation might be insightful both etically and emically. But, I am not convinced that, were they to repeat this experiment in nearby conceivable worlds, they would arrive at the same conceptual model, *unless* (i) they are grounding their explanation in something other than the data itself or (ii) they believe (and have shown) that there is a *systematic* reason we see these codes and these causal connections between them.

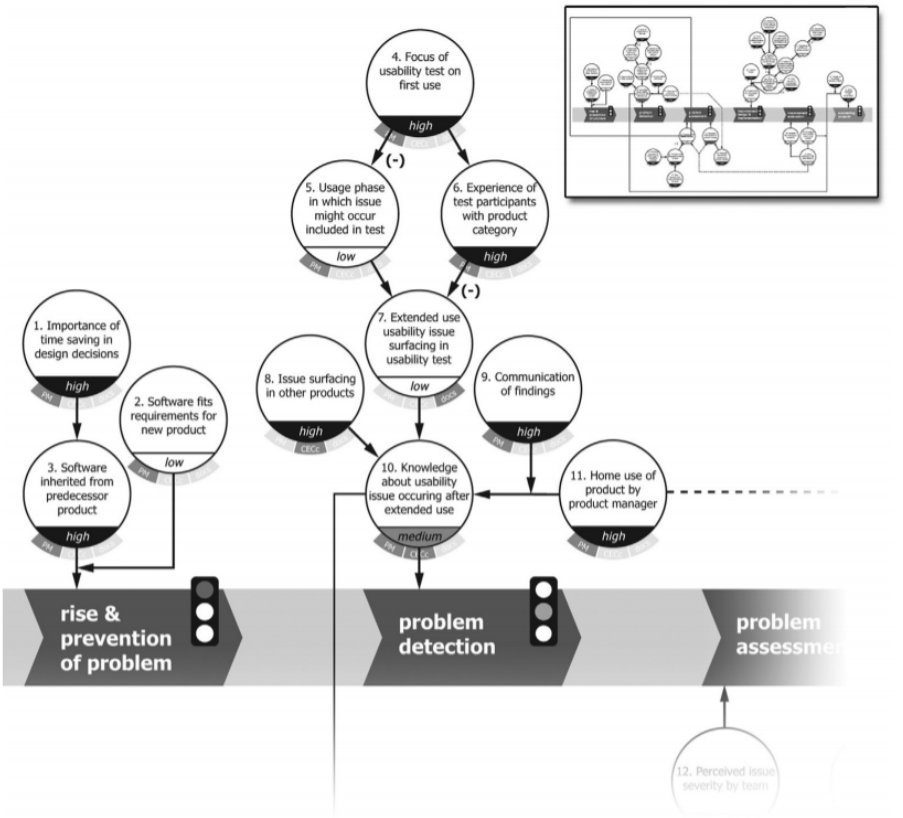

Figure 3: Example from Van Kujik, Jasper et al. "Drivers of Usability in Product Design." *Design Studies* 60. 2019.

Second, in Figure 4, the authors have projected a simpler high dimensional H (instances of codes grouped by speaker) down to a 2-dimensional plot. Like

above, this plot was produced from the author's intuitive understanding of the data. Unlike above, the plot shows the positions of speakers, not positions of variables. And further unlike above, the plot is a *metric* space. We are meant to believe, as viewers of this plot, that Mayne spoke similarly to Architect A, but in a "halfway down" sense along the y-axis; and that Mayne was *almost* the same as Hadid along that axis, but not quite. So here, order and measure matter in the visual and in the authors' conceptual explanation. I am more ready here than I was above to believe that this explanation would persist across nearby conceivable worlds. However, as David says in the QE book, where is the line?

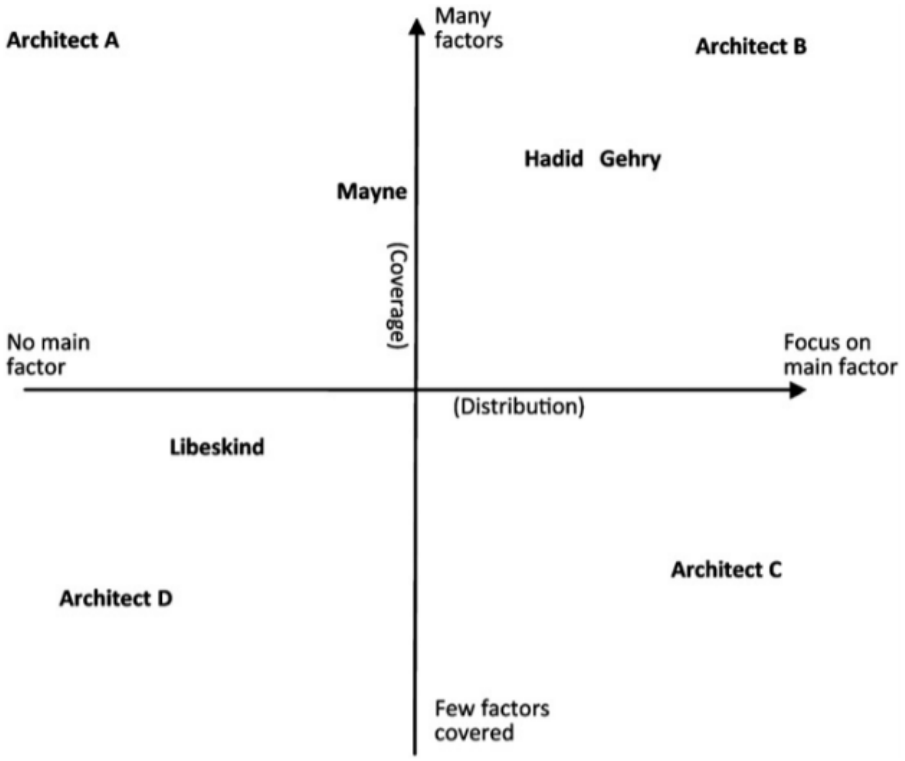

Figure 4: Example from Wang, Joseph F. "The Text of Free-Form Architecture." *Design Studies* 31. 2010.

Ethnographic explanations of events might hinge on ordinal and/or metric conceptions. Hopefully, those conceptions are correct and insightful, which the researcher is left to show theoretically. Whatever the explanation though, the researcher should actually believe it themselves. Member-checking, peer-checking, and analytical reflection can help with this. There is a problem though when one has an explanation like Van Kujik's that attempts to explain a complex model with a complex variable (eg, 6-buckets of time vs. a simpler 2-bucket before-and-after model). The problem is that the *explanation* cannot be con-

vincingly grounded in the data without a *lot* of data. What we are doing in hENA is providing a tool for modeling complex and/or sensitive tests like this.

So, geometrically, we can model our data as a high dimensional space H and rotate/project that space to some plane that models one or more of our explanatory variables. This allows a plot that is more directly grounded in the data instead of produced solely from researchers' intuitions (no matter how right those intuitions might be). And importantly, that plot will be, at the same time, aligned with the ordinal/metric facets already present in the researchers' ethnographic explanations.

From there, statistically, we can find evidence of saturation and effect size by running tests on the resulting positions of the data in those more grounded plots. In ENA, we project H by performing a rigid body rotation and we test for significance by a t-test. But, there is nothing holding a researcher to these approaches, so long as they can defend their choices.

The rigid body rotation has worked well enough for us in ENA, and it has its advantages.

5 Rotations

5.1 Top-Level Rotation Types

There are three types of rotations I have explored so far in hENA. In all three, we use a rigid body rotation and our goal is to find a plane that highlights some feature and project H to that plane.

First there is the SVD rotation, which highlights the two dimensions that explain the highest variance of H. We have been doing this in ENA for a while now.

Second, there are F1 rotations and the rotations that inherit from them. All F1 rotations highlight some variable of interest along the x-axis and show the dimension that explains the highest remaining variance along the y-axis. This is what was already done within ENA through the means rotation, MR1, which highlights the effect of a dummy grouping variable along x and uses an ortho SVD procedure to determine y. There is nothing preventing the variable of interest we model with the x-axis in an F1 rotation from being a continuous variable, a variable with six ordinal groups (like in Van Kujik), or so on, so long as it is an ordinal variable.

Third, there are F2 rotations and the rotations that inherit from them. All F2 rotations find two vectors in the high dimensional space, one for a first variable of interest and one for a second variable of interest. Once an F2 rotation finds these two vectors, it finds the plane where they are located, rotated such that the variable for the x-axis lies exactly on the x-axis, and the variable for the y-axis lies as close to the y-axis as possible. If these two vectors are already orthogonal or nearly orthogonal, then no problems occur. But if they are nearly colinear, then the y-axis will be uninterpretable. Regardless, an F2 rotation finds its plane by taking the first vector of interest to be the x-axis, and by

taking the rejection of the second vector from the first as the y-axis. This ensures that the plotted axes chosen are orthogonal.

5.2 Regression Framework

How the vectors are chosen in an F1 or F2 is up to the researcher. Generally, I have used a regression such as

$$H_i^T \sim 1 + G + K + GK \tag{7}$$

where G and K are two mean-centered variables of interest, G will be used to find the x-axis, and K will be used to find the y-axis. This I implemented in my initial R version of my approach during the QE class, and this facet of my approach has not changed since then.

The goal in using regressions is to find the effect of some variable on the grand mean of each dimension of H.

We can do this by, first, running a regression on each column i of H, treating H_i^T as the dependant variable and some variable of interest G as an independent variable. As we do, we jot down the coefficients for G for each regression, ie, β_{Gi}.

Since β_{Gi} represents the strength of G along that dimension of H, taken together, $\vec{B} = (\beta_{G1}, \beta_{G_2}, \ldots, \beta_{GN})$ represents the direction and strength of the overall effect of G on H.

An intuition of this is that, if G were a dial, and we were to turn it up, we would see a gradual and linear change, on average, to the networks (in ENA) or column values (in C&C); the direction $\vec{B}$ models this linear change. Put another way, $\vec{B}$ is the *account* of the networks given by G.

Second, we normalize $\vec{B}$ such that it has a unit length and we use that as our x-axis. In F1, we take an ortho SVD of $\vec{B}$ to find the y-axis. And in F2, we find a second vector of interest using the same method as above, say $\vec{C}$, we reject $\vec{C}$ from $\vec{B}$, we normalize the result, and we use that as the y-axis.

To interpret the resulting axes, in both F1 and F2, we say that the x-axis models the average account of the networks/column values given by the variable of interest. In F2, we say that the y-axis models the average-and-independent-of-x account of the networks/column values given by the second variable of interest.

5.3 Family Tree

Although the regression approach has not changed, my interpretation of the approach has, creating a family tree of rotation types.

A means rotation (MR1) is just a special case of an F1 rotation. The logic is the same, just using a dummy variable G where 0 represents one group and 1 represents another group. And visually, it makes sense to plot the CIs for the group mean positions in an MR1, whereas in F1 generally there are no groups to talk about. So the logic for dealing with MR1s in jENA can wrap around

the logic for F1's in general: We create the dummy variable up front, and we append the CIs to the plot at the end.

And double means rotation (MR2) is just a special case of an F2 rotation, using a pair of dummy variables instead, one prioritize for the x-axis. And as we did for the ICQE paper, we can break the plotted CIs down further to show the interaction between these two groups, resulting, as we have done it, in an "elevator plot." And again, this is just a wrapper around the logic of what is already being done for F2.

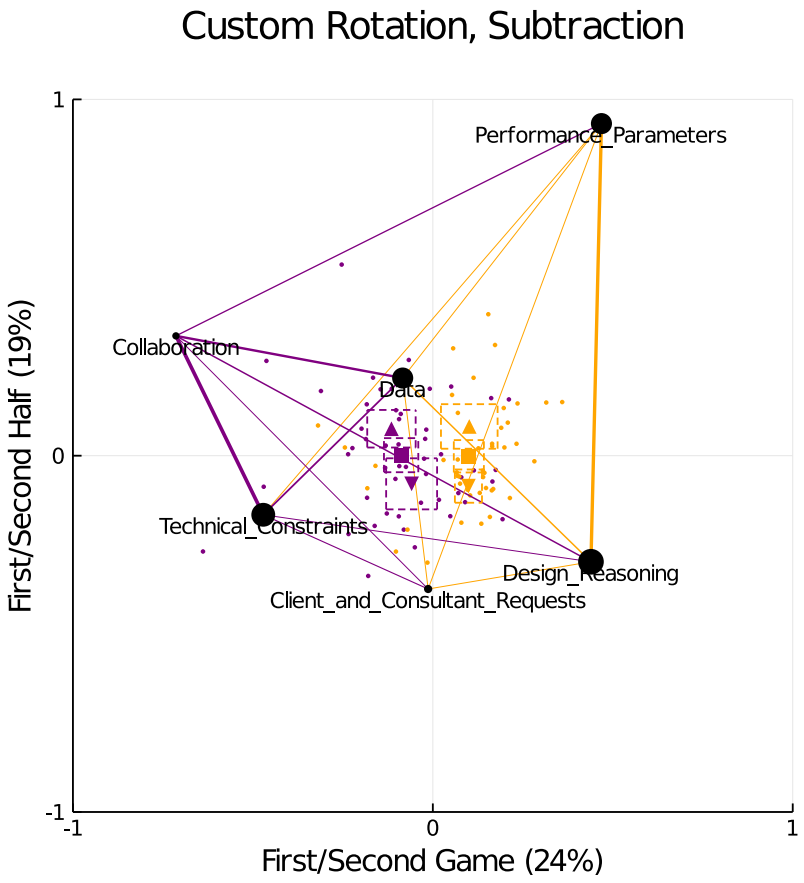

Figure 5: Example MR2 Rotation. First Half and Second Half subgroup means are shown with ▾ and ▴ respectively.

5.4 Controls etc.

Regardless of the particular subtype of F1 or F2, the rotation process can be seen as an optimization procedure. It finds a plane such that the resulting distribution of the variable of interest is as close as possible (within that plane) to what its distribution would be in one's ideal, geometric metaphor (the ordinal/metric facets of their conceptual explanation).

Controls on the regression, such as holding other variables constant or controlling for nesting in teams—these are just adjustments to that rotation optimization process. In other words, we could say that we rotated to give the closest match to our geometric metaphor *while holding such and such constant along the x-axis*, or so on. One could even replace the linear regression with a LASSO regression, which would produce the best fit with one's geometric metaphor while punishing variables with low correlation to the variable of interest. Figure 6 gives an example of this. I have a hunch, and I explored this a little in one of my QE paper drafts, that a LASSO regression would produce an ENA model that would better pass cross-validation. If nothing else, it produces a lower-dimension model.

Whatever the particulars, including controls in the rotation step allows us to produce a visual *that* there is a pattern and *what* the pattern is *while holding such and such constant along the x-axis*. Getting at this was my initial inspiration in this practicum: Yes, we can run ENA without controls, get plotted points for an axis, and test with a post hoc regression that the groups differ along this axis even when controlling for such and such. However, the evidence *that* there is a pattern would come from the regression; the explanation of *what* that pattern is would come from the ENA plot, which did not include controls in its model. This is a problem because there becomes a disconnect between the evidence and the explanation.

6 Visualization

6.1 Attempts so Far

In order to interpret the patterns in the high dimensional space, especially in terms of an axis derived from a variable of interest like in F1 and F2, visualizations should show:

1. The distribution of units across the space
2. The distribution of codes/connections across the space
3. The distribution of variables of interest across the space
4. The dynamics of change in those codes/connections as one moves through the space along some axis of interest
5. The dynamics of chance in those codes/connection as one moves from one group of units of interest to another

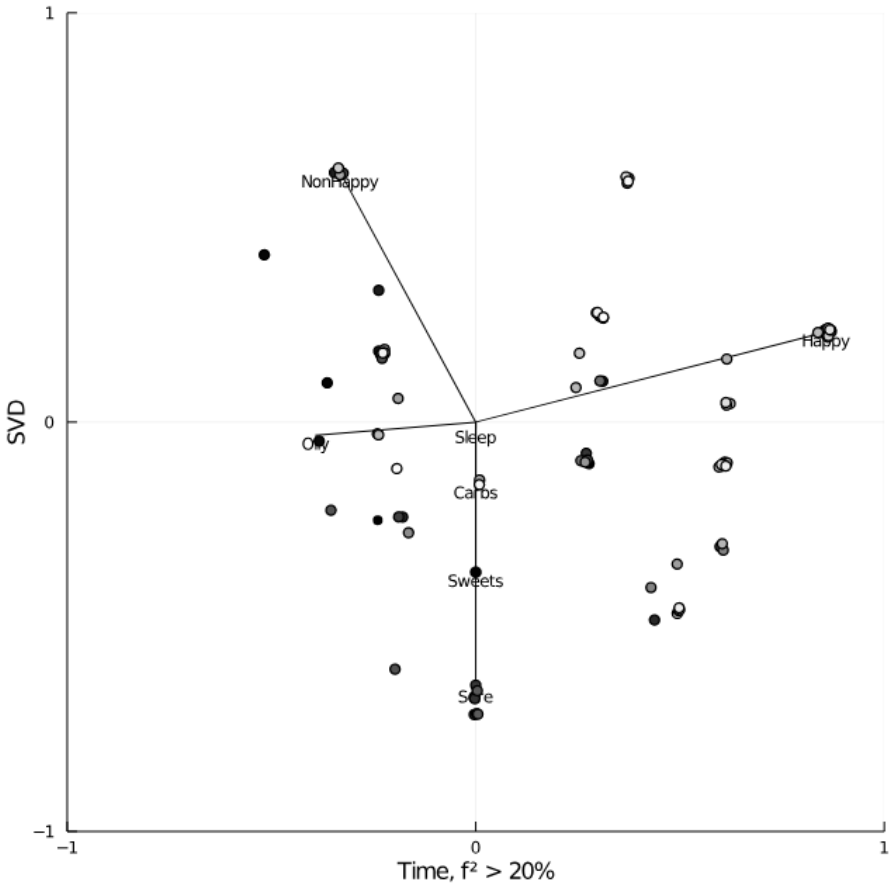

Figure 6: Same plot as Figure 1, but computed using a LASSO regression instead of a linear regression. NonHappy, Oily, and Happy have hardly moved, but the remaining codes have been collapsed to zero along the x-axis. This reduces the variance explained by Time from 23% to 20%, but it produces an alternative account of the relationship between Time and the codes that is simpler and more likely (I believe) to pass cross-validation.

I am convinced that a single plot cannot show all of these, and so my implementation in jENA produces a multi-plot. This makes it easier for the programmer to get all the plots they may need, it forces them to produce them all, and it prevents them from producing one type of plot at a time.

What those plots should contain is a topic of current discussion.

I have tried and liked the following:

- Meet goals 1 and 2 by showing a plot with the units and overall total code/connection weights (an "omnibus plot"). We currently do this in

SVD rotations, but I am doing it for MR1 rotations too.

- Meet goal 4 by generalizing the subtraction plot that we currently use in MR1 rotations. To do this, I find, for each code/connection, the slope and correlation between units' x-axis positions and their individual weights for that code/connection. The absolute value of the slope is used to determine the width of the plotted code/connection, and the correlation is used to determine the color of the code/connection. I have used red for negative and blue for positive. At Cesar's advice, I have used a nonlinear gradient such that stronger correlations have a more saturated color, while weaker correlations have a less saturated color. The purpose of this is to visualize the strength of a relationship, since, for a strong effect size like we typically see in MR1, the generalized subtraction plot is visually identical to the status quo subtraction plot. But the effect sizes of different continuous variables should be distinguishable visually. Thus, the correlation-saturation trick.

- Meet goal 5 as we currently do in ENA. We do this by plotting only the units within that group and determining the thickness of the plotted codes/connections using only that group. Visually, these widths should be less than the width of the omnibus plot, since all the group plots should "sum" to the omnibus plot.

This produces $N + 2$ plots, where N is the number of groups to show: one plot for the omnibus, one plot for the generalized subtraction plot, and one plot for each group.

Meeting goal 3 has been the hardest. I have tried two approaches, and I'm sold on neither.

The first is to color-code the units to represent their value for the variable of interest. For example, color-code student units based on their grade on a test, when we are modeling H with an F1 rotation that uses test grades for the x-axis. The problem with this is that it is hard to decide on a color scale that works regardless of the distribution of the variable of interest. What should we do for normally distributed data? bi-modal data? data with a long tail? etc. We are also not sure whether the "middle" of the scale should be the median or the mean of the variable of interest, or something else.

The second is to plot a histogram showing how the variable of interest is distributed in the space. Imagine each unit dripping a drop of paint based on its color-coding, as done above, into a bucket below it. The liquid-level inside the bucket represents how many units are above it, and the color of the paint in the bucket represents the average "color," or value of interest, of those units. When using a strong effect like we see in MR1, the resulting histogram shows a clear separation of the groups in terms of the values of interest, ie, their dummy group variable value of 1 or 0. However, for a continuous variable, just as above, it is unclear what color scheme would work best for what distributions, and the results tend to appear much more muddied. I have tried to tweak the color

scheme to visually distinguish three test cases (a strong, medium, and weak effect size), but this is far from solved.

Figures 7 and 8 illustrate these approaches.

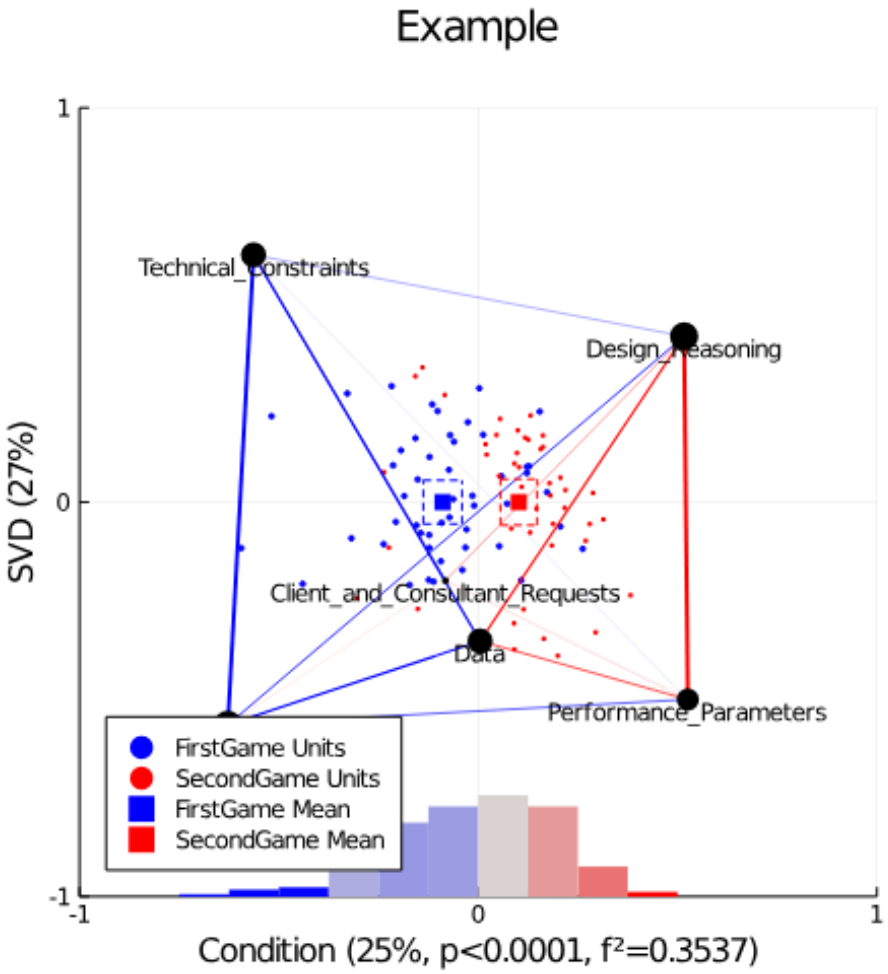

Figure 7: Example ENA plot showing color-coded dots, a status quo subtraction plot, and a color-coded histogram.

6.2 Remaining Issues

In a model like Van Kujik's, there are six buckets to be placed along the x-axis. Even if these do occur magically in a linear pattern in the high dimensional space, the ENA plot will still be hard to separate into six distinct groups.

The same problem occurs when we use a continuous variable, like test grades, for the x-axis. The low and high scores will be distinguishable. But the "average

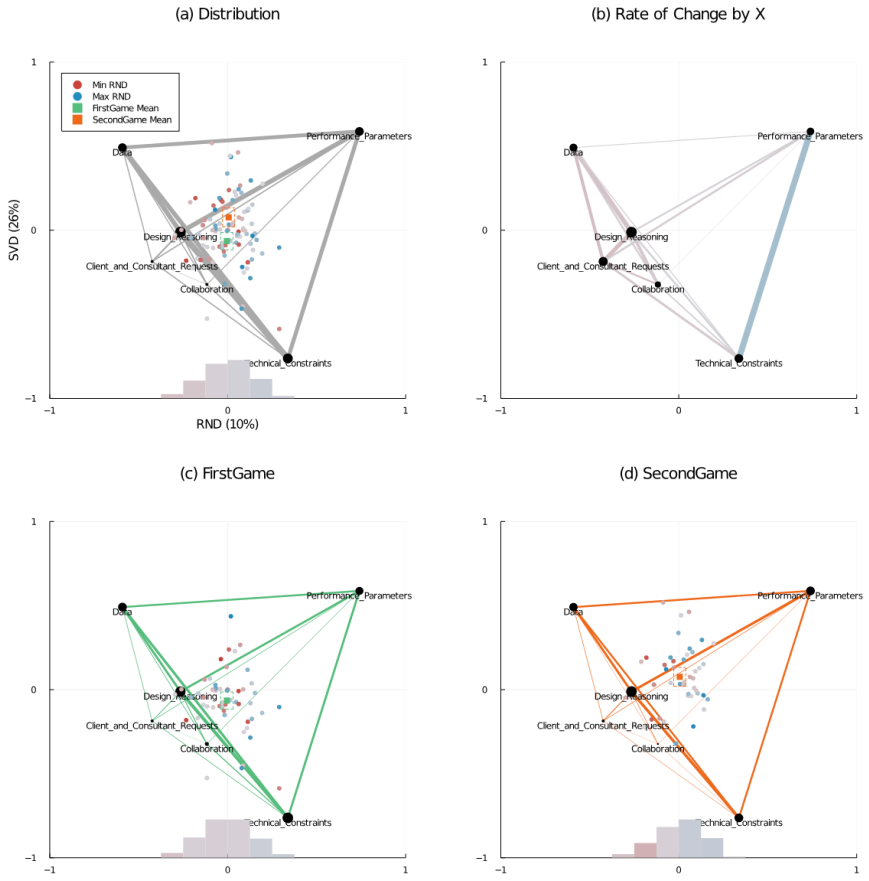

Figure 8: Example ENA multiplot, showing (a) the omnibus distribution of units in the space and a muddied histogram of a variable (randomly-assigned values) that gives a weak account of the networks; (b) a generalized subtraction plot showing the "washing out" effect from using less saturated colors for weaker correlations; (c-d) grouped versions of the omnibus plot

overall" connections will not be distinguishable from the "middle test scores" connections.

7 jENA API

7.1 Minimal Examples

Listing 1 gives a minimum working example of the Julia implementation of ENA ("jENA"), as of the time of writing. The live version of the code is hosted on github at `github.com/snotskie/EpistemicNetworkAnalysis.jl`. In

this example, the RescueShell data (packaged with jENA) is loaded, fit to a default ENA model, and displayed as a text-based object. This will print the DataFrames that make up the ENA model and coregistration scores.

```
using EpistemicNetworkAnalysis

data = ena_dataset("RS.data")
codes = [
    :Data,
    :Technical_Constraints,
    :Performance_Parameters,
    :Client_and_Consultant_Requests,
    :Design_Reasoning,
    :Collaboration
]
conversations = [:Condition, :GameHalf, :GroupName]
units = [:Condition, :GameHalf, :UserName]

ena = ENAModel(RSdata, codes, conversations, units)
display(ena)
```

Listing 1: Minimum Working Example

In order to run that example, one first needs to install the `EpistemicNetworkAnalysis` package from github. Listing 2 shows this.

```
using Pkg
Pkg.add("https://github.com/snotskie/EpistemicNetworkAnalysis.jl")
```

Listing 2: Installation Command

To display an ENA plot, first create a plot object from the ena object, then call `display` on the plot object. Listing 3 shows this.

```
p = plot(ena)
display(p)
```

Listing 3: Minimal Plotting Example

There is no way to change the rotation of an existing model currently in jENA. Instead, create a rotation object to hold the configuration of that rotation and pass that object into a new call to the ENA model constructor. Listing 4 shows this.

```
rotation = MeansRotation(:Condition, "FirstGame",
    "SecondGame")
ena = ENAModel(RSdata, codes, conversations, units,
    rotateBy=rotate)
```

Listing 4: MR1 Example

7.2 Options

At the time of writing, one can configure rotations, the ENA model, and the plot function. Respectively, Listings 5–7 show kitchen sink examples for these.

```
# Default
SVDRotation()

# Grouping variable, left group, right group
MeansRotation(:Condition, "FirstGame", "SecondGame")

# Model type, coefficient index, formula, contrasts dict
FormulaRotation(
    LinearModel, 2, @formula(y ~ 1 + CONFIDENCE_Pre), nothing
)

# x-axis config
# y-axis config
Formula2Rotation(
    LinearModel, 2, @formula(y ~ 1 + CONFIDENCE_Pre), nothing,
    LinearModel, 1, @formula(y ~ 1 + 0), nothing
)
```

Listing 5: Rotation Options

```
ENAModel(
    # Required
    data,
    codes,
    conversations,
    units,

    # Optional, defaults shown
    windowSize=4,
    rotateBy=SVDRotation(),
    sphereNormalize=true,
    dropEmpty=false,
    deflateEmpty=false
)
```

Listing 6: ENA Model Options

```
plot(
    # Required
    ena,

```

```
    # Optional, defaults shown
    margin=10mm,
    size=600,
    lims=1,
    ticks=[-1, 0, 1],
    titles=[],
    xlabel="X",
    ylabel="Y",
    leg=:topleft,
    negColor=DEFAULT_NEG_COLOR,
    posColor=DEFAULT_POS_COLOR,
    extraColors=DEFAULT_EXTRA_COLORS,
    flipX=false,
    flipY=false,
    groupBy=nothing,
    showExtras=true,
    showNetworks=true,
    showUnits=true,
    showCIs=true
)
```

Listing 7: Plot Options

7.3 Infrastructure

jENA makes use of two features of Julia.

The first is a composed type hierarchy. While not currently implemented, it should be possible for one to define a `DirectedENAModel` type and inherit as much as possible from `ENAModel` (or, refactor and inherit the other way).

There is not, however, a single `ENAModel`. It is better to think of the `ENAModel` type as a *parametric type*. For example, an ENA model using an SVD rotation vs. one using an MR1 rotation have a lot of logic in common. However, what we ought to do when we plot them differs. Therefore, the first is represented by the type `ENAModel{SVDRotation}` and the second by `ENAModel{MeansRotation}`. In this way, the latter can inherit logic from the first, but make adjustments where needed.

This leads into the second feature, Julia's "multiple dispatch" system, which allowed me to conceptualize the internal API calls inspired by Aspect-Oriented Programming.

Each rotation type implements a call to the `rotate!` function, which is a helper called by the ENA model class. Listing 8 gives examples.

```
# SVD
function rotate!(rotation::AbstractSVDRotation,
    networkModel::DataFrame, unitModel::DataFrame,
    metadata::DataFrame)
```

```
3
4       ## Run an ortho svd and use those values as the axis
        weights
5       pcaModel =
        projection(help_deflating_svd(networkModel,
        unitModel))
6       networkModel[!, :weight_x] = pcaModel[:, 1]
7       networkModel[!, :weight_y] = pcaModel[:, 2]
8   end
9
10  # MR1
11  function rotate!(rotation::AbstractMeansRotation,
        networkModel::DataFrame, unitModel::DataFrame,
        metadata::DataFrame)
12
13      ## Manually factor the grouping variable to
        0/1/missing
14      metadata[!, :FactoredGroupVar] =
        map(eachrow(metadata)) do unitRow
15          if unitRow[rotation.groupVar] ==
        rotation.controlGroup
16              return 0.0
17          elseif unitRow[rotation.groupVar] ==
        rotation.treatmentGroup
18              return 1.0
19          else
20              return missing
21          end
22      end
23
24      ## Use a FormulaRotation to do the rest of the work
25      invoke(rotate!, Tuple{AbstractFormulaRotation,
        DataFrame, DataFrame, DataFrame}, rotation,
        networkModel, unitModel, metadata)
26  end
```

Listing 8: Example `rotate!` Functions

Similarly, the `plot` function calls a number of helpers, each of which can be overridden at any point in the composed type tree. Listing 9 gives one example of this. Here, the `plot_units!` function is defined for any ENA model using an MR1 rotation. This function would also be called on any sub-type that inherits from this point in the composed type tree.

```
1   function plot_units!(p::Plot,
        ena::AbstractENAModel{<:AbstractMeansRotation},
        displayRows::Array{Bool,1};
```

```
    flipX::Bool=false, flipY::Bool=false,
    minLabel::Union{Nothing,String}=nothing,
    maxLabel::Union{Nothing,String}=nothing,
    kwargs...)

    ### Use meaningful legend labels for the units
    if isnothing(minLabel)
        minLabel = "$(ena.rotation.controlGroup) Units"
    end

    if isnothing(maxLabel)
        maxLabel = "$(ena.rotation.treatmentGroup) Units"
    end

    ### Let formula rotation do the rest of the work
    invoke(plot_units!, Tuple{Plot,
    AbstractENAModel{<:AbstractFormulaRotation},
    Array{Bool,1}},
        p, ena, displayRows; flipX=flipX, flipY=flipY,
    minLabel=minLabel, maxLabel=maxLabel, kwargs...)
end
```

Listing 9: Example `plot` Helper Function

7.4 Model Computation

In jENA, an ENA model is composed of multiple DataFrame objects: one for the original data, one for the accumulated totals per unit, one for the corresponding metadata, one for the correpsonding centroid positions, one for the axis weights of each code connection used to compute the rotation, and one for the codes' positions in the high dimensional space.

After initializing these structures with placeholders, the `ENAModel` constructor first accumulates code counts for each unit, as we currently do in rENA.

Second, units are sphere normalized.

Third, if the option is set, empty units are dropped.

Fourth, the position of codes in B are computed, as described in Equation 2.

Fifth, the code positions are used to compute the centroid positions in the high dimensional $\hat{H}$.

Sixth, a high dimensional space is prepared for computing the rotation vectors. By default in jENA, this is just the centroid space $\hat{H}$, but this could in theory be swapped with an optional argument with the accumulated space H. I have not implemented that yet, though it would be simple. Currently, one can compute rotations on the deflated space $\ddot{H}$ by setting `deflateEmpty=true`.

Seventh, the positions of the units in H and $\hat{H}$ along the axes described by the rotation vectors are computed. The same are done for the code positions.

Finally, a test is performed on the rotation vectors to ensure that they are orthogonal. Since the rotation methods could be user-defined, this is an assumption check within the `ENAModel` constructor itself. If this test fails, a warning is raised.

There are two immediate and notable differences in this from rENA:

- Rotations are performed on the centroid space by default
- The order of operations is Optimize-Rotate, not Center-Rotate-Optimize

8 CUSUM

I'm toying with this idea. I haven't run it by David yet, and I've chatted about it briefly with Ariel.

In Figure 6, Oily and NonHappy are to the left, and Happy is to the right. There is a clear relationship between those codes and time, which, sure, is true empirically. But the qualitative features of the structure of that relationship is missing. Closing the interpretive loop, I know that my skin cleared up nearly all at once (perhaps due to the Summer ending, or due to being on Estradiol long enough, or some combination). I also know that while I became less anxious over time during the period reflected by Figure 6, it wasn't too much of a change; the bigger change was that I greatly and gradually began marking "happy" each day.

So, these three codes have three different structural relationships with time, these distinctions qualitatively matter, and the biplot does not show this. ENA plots with a time-based F1 rotation would have the same problem. We want to tell a story where the relationship between the continuous variable and the codes matter, but this relationship is not a "one and done" sort of thing. We can say that there are negative/positive effects but...so what? To really close the interpretive loop in an F1 rotation we need to go a step further. I'm not sure what that step needs to be generally, but for time-based variables, I have an idea for how we can model structural shifts in our qualitative codes as we "walk" through the data over time: a CUSUM framework.

A CUSUM framework can capture and test for those structural shifts. Perhaps others can adapt this to work with trajectories. And perhaps others can find a way to use this or build on this to work with any continuous variable. I do know that, right now, a limitation is that it assumes even time-steps: if data is missing or irregularly collected in time, then the CUSUM framework I describe here doesn't quite work.

Figure 9 shows the idea. By using CUSUM plots of dimensions of interest (individual codes in C&C biplots, or individual connections in ENA), we can see how the frequency of those codes changed over time. We can also see evidence that structural shifts have occurred. By that, I mean that in CUSUM stability testing, we assume some model and test for structural shifts from that model. In Figure 9, the assumption is that there is a frequency at which each code occurs and that frequency does not change over time.

For HAPPY and OILY we see that static frequency is not the case, and we see the nature of the relationship between those codes and time: in line with the qualitative interpretation, HAPPY gradually increases and OILY suddenly drops off. For NONHAPPY, there does appear to be a negative slope in its intercept over time, but not enough that an intercept-only model can't keep up, even though there is just enough of a relationship between time and NONHAPPY that NONHAPPY was not collapsed by the LASSO in Figure 6.

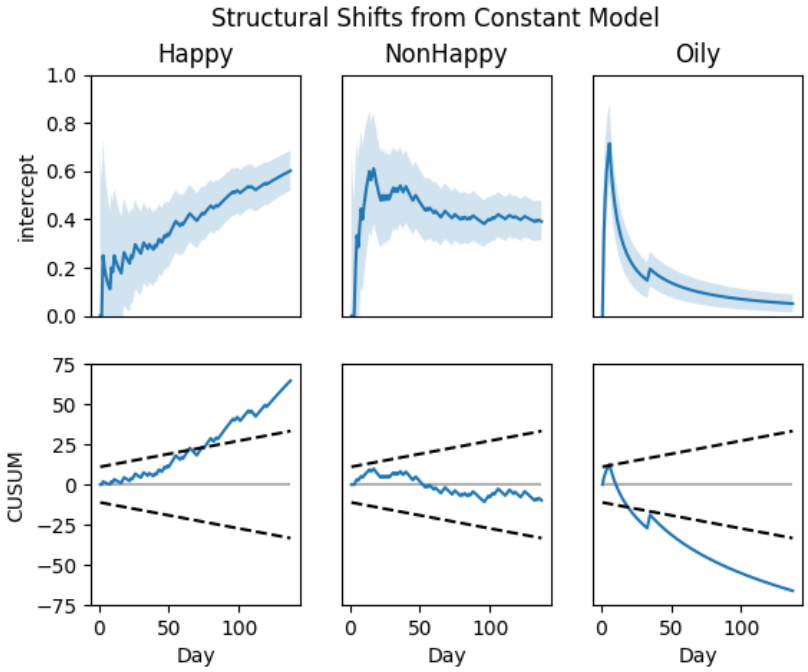

Figure 9: Example CUSUM plot of the same data as Figures 1 and 6. The top row shows the expected value of each code as time progresses under an intercept-only model. Note that the intercept for HAPPY continuously rises, the intercept for NONHAPPY continuously falls at a lower rate after a period of model noise, and OILY suddenly falls off. The second row shows the cumulative sum of residuals (CUSUM) over time and a $\alpha = 0.05$ interval where the CUSUM is expected to be under the null hypothesis that there are no structural shifts. Where the CUSUM escapes the interval shows when and in what direction a structural shift has occurred. As time goes on, more data is collected, and the underlying system levels out again, it is possible for a CUSUM to return to the interval again.