Assignment 4 - Second Steps

Joey Heffner
Jae-Young Son
James Wilmott

Overview

As a brief reminder, we are replicating portions of the study 'Maps of subjective feelings' by Lauri Nummenmaa, Riitta Hari, Jari K. Hietanen, and Enrico Glerean. This study can be found at: https://www.pnas.org/content/115/37/9198. The goal of this study was to elucidate organizing principles of subjective experiences of emotions across individuals, and to uncover whether/how emotions can be grouped together to form broad categories of subjective experience.

Our previous efforts replicated the findings from Experient 1, including replicating Figure 1 from the paper. As a brief reminder, participants were presented with 100 words that corresponded to core bodily or emotional states and asked to rate them according to 5 dimensions that the authors pre-determined were of interest. For a more in-depth description of that experiment and our efforts to replicate their analysis, please check our previous assignment.

For this assignment, we will continue our efforts at replicating portions of the study with Experiment 2. Here, the authors assessed the similarity of emotional and bodily states across individuals. Participants were asked to make similarity judgments between pairwise combinations of 100 feelings. The authors then extracted pairwise similarities between words by determining the (scaled) Euclidean distance between each possible set of words, thereby creating an average similarity matrix across participants for each of the pairwise ratings. These similarities across participants were highly reliable (mean Spearman's correlation split-half reliability of 0.94 across 5000 random iterations, Figure S3). The mean distance matrix was computed by averaging across participants, and therefore represents the group-level degree of similarity between each of the 100 words.

Using this similarity matrix, the authors set out to determine whether unsupervised learning techniques can reveal underlying clusters of emotions, which represent structure in how people experience emotional states. To do so, they used the DBSCAN clustering algorithm (density based spatial clustering of applications with noise), which resulted in 5 distinct clusters of states, which they termed the following:

1. Positive emotions
2. Negative emotions
3. Illnesses
4. Homeostasis
5. Cognition

Other words that were not assigned to these clusters were labeled 'Unspecific'.

To graphically communicate how the 100 words clustered together, the authors used a dimensionality reduction technique known as t-SNE (t-distributed stochastic neighbor embedding), which is *conceptually* akin to a nonlinear PCA (principal components analysis). Briefly, the goal of t-SNE is to represent similarity/dissimilarity in high-dimensional space by placing points closer together/further apart in 2D space. Importantly, the use of t-SNE in this experiment was *not* an analytical technique, but instead a convenient method for visualizing similarity/dissimilarity in high-dimensional space. The authors then colored the points according to the clusters identified by the DBSCAN algorithm. The ultimate result of this visualization is Figure 2. Therefore, Figure 2 is a two-dimensional representation of "feeling space", which is essentially a map between feelings arranged by t-SNE and clustered with DBSCAN.

Here, we attempt to replicate the DBSCAN clustering analysis and t-SNE visualization displayed in Figure 2.

DBSCAN

As a first pass, we use DBSCAN on the similarity matrix and visualize it using PCA (a linear dimensionality reduction technique). DBSCAN is a density-based clustering algorithm which can identify irregularly shaped clusters in datasets with noise or outliers. Compared to other unsupervised machine learning approaches (e.g., K-means), DBSCAN has the following advantages: 1) it does not require a specific number of clusters, 2) it can find irregularly-shaped clusers, and 3) it can identify outliers.

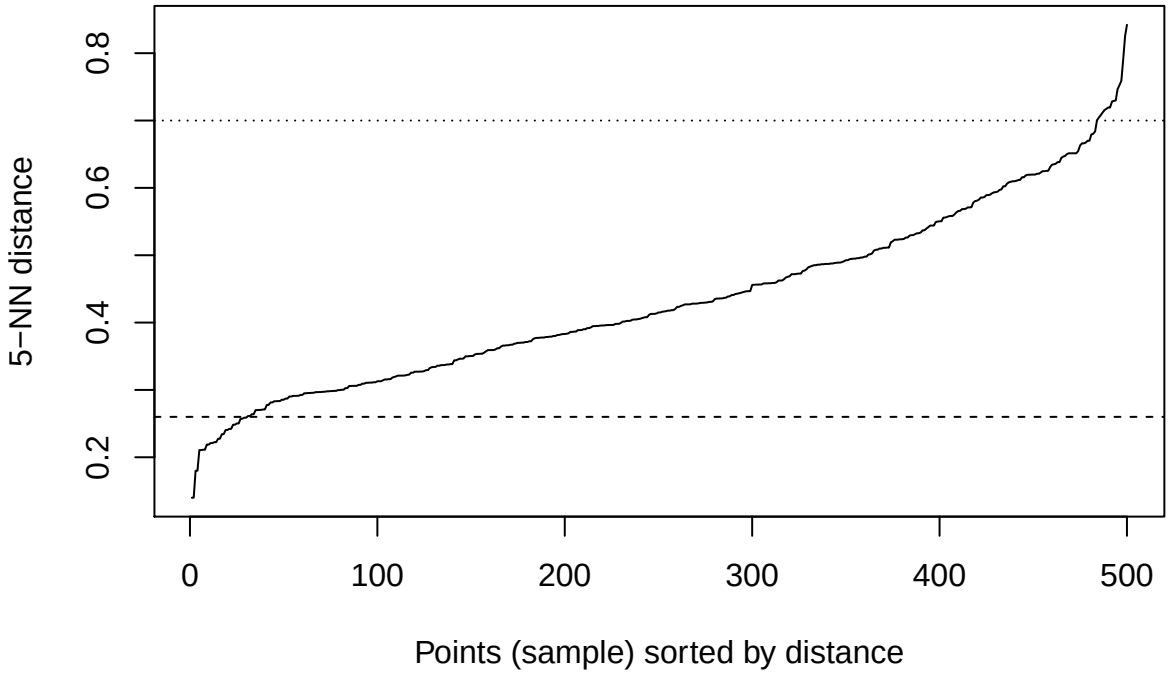
Notably, the authors of this study implemented DBSCAN using MATLAB. If their analysis is generalizable across different software implementations, we would expect that implementing DBSCAN using R should lead to the same cluster results as the original paper.

DBSCAN parameter selection

There are two parameters for DBSCAN:

1. **epsilon** (`eps`): defines the radius of a neighborhood around a specific point
2. **minimum points** (`MinPts`): defines the minimum number of neighbors with "eps" radius

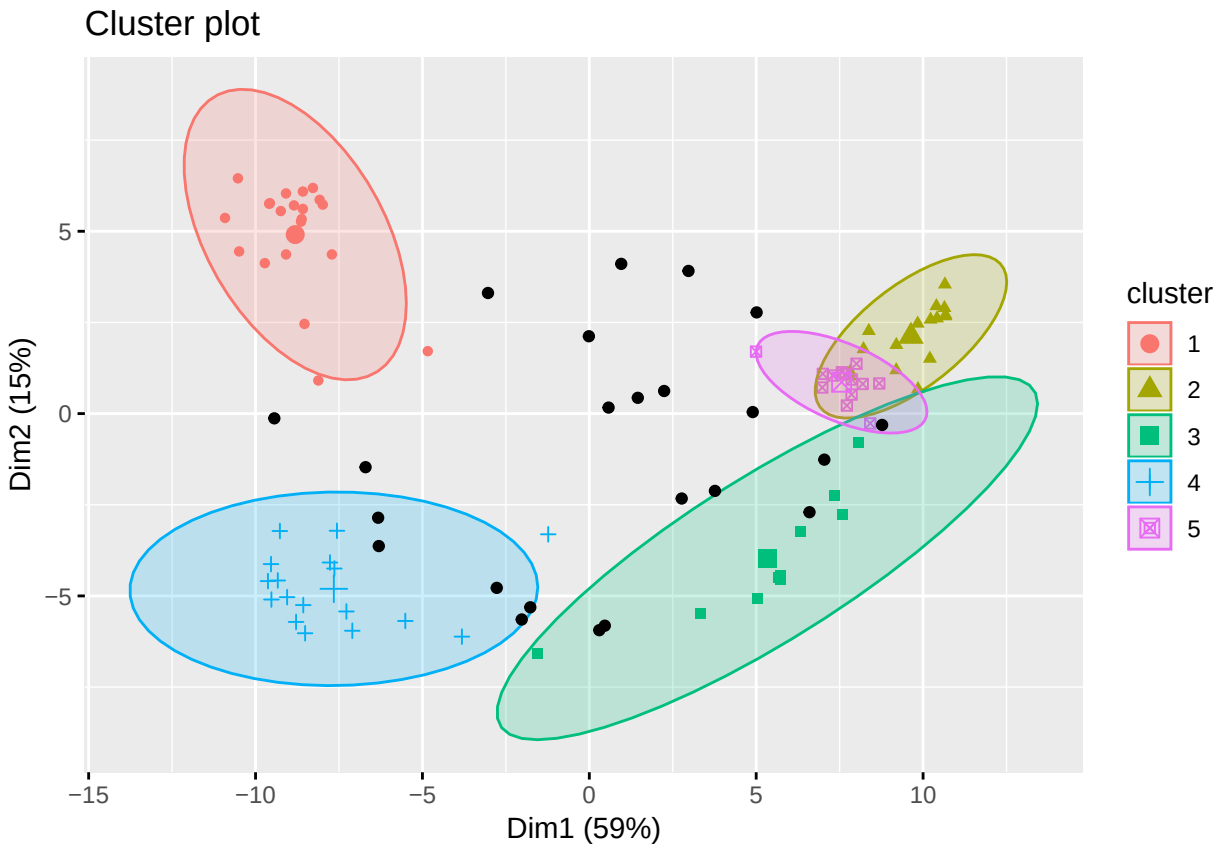
First, we search for the optimal value of `eps` using the k-Nearest Neighbor Distances in the similarity matrix using `MinPts=5` (the value used by the authors). We can do this by searching for where there is a "knee" in the plot.



The original analysis used `eps=0.2565`, which approximately accords with the graphical result visualized above. Now that we have values for both DBSCAN parameters, we can run the analysis.

DBSCAN cluster analysis

As the plot below demonstrates, we are able to replicate the original finding that there are five unique clusters.



However, we do not yet know how similarly specific sensations are clustered in the R implementation of DBSCAN versus the MATLAB implementation. To find out, we can simply correlate our DBSCAN results against the original results.

The correlation between our DBSCAN results and the original DBSCAN results is 0.833189. Notably, this is despite us using the exact `eps` and `MinPts` parameter values as the original analysis. What causes this difference? The documentation for the `fpc::dbscan` function notes that "this is a simplified version of the original algorithm (no K-D-trees used), thus we have `o(n^2)` instead of `o(n*log(n))`". This discrepancy serves as a good cautionary tale for cross-platform replication attempts: software packages often implement the same *conceptual* functions using different algorithms, and it is not always transparent when software developers differ in their design choices.

Visualizing clusters using t-SNE

The authors' dataset includes the cartesian coordinates of each sensation from their t-SNE analysis implemented in MATLAB. Due to the complexity of this technique, In the figure below, the top panel is an **exact** replication of Figure 2 from the original paper, meaning that it uses the MATLAB implementation of DBSCAN. The bottom panel visualizes our replication, meaning that it uses the clustering results from the R implementation of DBSCAN. Although results are largely consistent, there are a few differences. Namely, hunger, pride, and tiredness were not able to be classified by R's version of DBSCAN.


OriginalDBSCAN
OurDBSCAN
t−SNE2
t−SNE1
Losing
Ostracism
Sadness
Failing
Loneliness
Fear
Shame
Despair
Depression
Anger
Despise
Guilt
Anxiety
Hostility
Disgust
Panic
Stress
Exhaustion
Nervousness
Tiredness
Mania
Acceleration
Craving
SexualArousal
Wanting
Orgasm
FeelingTouch
Pride
ClosenessInSocialRelations
Winning
Happiness
Togetherness
Love
Laughing
SocialLonging
Sympathy
LongingFor
Vitality
Relaxation
Heartbeat
Movement
Daydreaming
Breathing
Defecation
Numbness
Suffocation
Shivering
Sweating
Dizziness
Coldness
Hotness
FeelingNauseous
HavingCold
HavingStomachFlu
Coughing
HavingFlu
Sneezing
HavingToothache
Aching
Itching
FeelingPain
Thirst
Hunger
Drinking
Eating
Tasting
Satiation
Smelling
Seeing
Hearing
Surprise
BeingConscious
Recognition
Imagining
Memorizing
Recollection
Reasoning
Remembering
Estimating
BeingDazzled
SelfRegulation
Forgetting
Waiting
DBSCAN Classification
Unclassified
Negative Emotion
Positive Emotion
Illnesses
Cognitive Processes
Homeostatic States

Plans for study extension

For the extentions portion of the project, our group will compare the results of alternative clustering algorithms to the original reported findings. The authors chose to use the DBSCAN clustering method to assess the underlying structure of the emotional spectrum. We will use alternative clustering algorithms in place of the DBSCAN method and compare the results of these clustering techniques to the authors' findings. Accomplishing this will tell us how dependent the authors' results are on their choice of analysis method, and whether different clustering techniques lead to different conclusions about the structure of emotional experiences.

Code Appendix

```
knitr::opts_chunk$set(echo = FALSE)

# Initialize environment
libraryBooks <- c("knitr", "tidyverse", "cowplot", "dbscan", "fpc", "factoextra", "Rtsne")
invisible(lapply(libraryBooks, require, character.only = TRUE)); rm(libraryBooks)

# Path to data
# scriptPath <- dirname(rstudioapi::getSourceEditorContext()$path)
scriptPath <- getwd()  # WHY does knitr complain about the interactive rstudio calls

# Graph aesthetics
pnas_theme = theme_bw(base_size = 10) +
  theme(text = element_text(size = 10), # Increase the font size
        panel.grid = element_blank(),
        axis.ticks = element_blank()) # remove x & y ticks

# Load data
d1 <- read.csv(paste0(scriptPath, "/Data/DissimilarityMatrixForFig2.csv"), header = TRUE)

# Determine optimal `eps` (look for 'elbows' in KNN plot with the same number of MinPts
dbscan::kNNdistplot(d1[,-1], k = 5) # remember to remove Row col
abline(h = 0.26, lty = 2)
abline(h = .7, lty = 3)

# DBSCAN RUN NOTES
# fpc has a dbscan() function which works like follows
# fpc::dbscan(data, eps, MinPts = 5, scale = FALSE, method = c("hybrid", "raw", "dist"))
# data = data matrix, dataframe, or dissimilarity matrix (specify method = "dist"). Euclidean distances
# eps = reachability maximum distance
# MinPts: Reachability minimum number of points
# scale: if TRUE data will be scaled
# method:
#   hybrid: expects raw data, calculates partial distance matrices
#   raw: expects raw data
#   dist: treats data as distance matrix

# DBSCAN clusters require a minimum number of points (MinPts) within a maximum distance (eps) around on
# any point within eps around any point which satisfies conditions is a cluster member (recursively)
# some points may not belong to any clusers and are treated as noise
```

```r
# Run DBSCAN
db <- fpc::dbscan(d1 %>% select(-Row), eps = 0.2565, MinPts = 5, method = "dist")
# print(db)

# Graph
fviz_cluster(db, d1 %>% select(-Row), geom = "point", ellipse.type = "norm")

# Predict class labels of our DBSCAN data
predLabels <- predict(db, d1) # specifies the cluster given to each 'feeling'
feelingNames <- colnames(d1 %>% select(-Row))

predDF <- data.frame(DBSCAN_labels = predLabels, feeling_labels = feelingNames)

# Load their DBSCAN results and make sure the sensation labels match
d2 <- read.csv(paste0(scriptPath, "/Data/Exp2Classifications.csv"),
               header = TRUE, stringsAsFactors=F) %>%
  filter(!is.na(DBSCAN_class)) %>% # remove na from bottom
  mutate(sensations = replace(sensations, sensations=="Being conscious", "BeingConscious"),
         sensations = replace(sensations, sensations=="Being dazzled", "BeingDazzled"),
         sensations = replace(sensations, sensations=="Closeness (in social relations)",
                              "ClosenessInSocialRelations"),
         sensations = replace(sensations, sensations=="Feeling nauseous", "FeelingNauseous"),
         sensations = replace(sensations, sensations=="Feeling pain", "FeelingPain"),
         sensations = replace(sensations, sensations=="Feeling touch", "FeelingTouch"),
         sensations = replace(sensations, sensations=="Having cold", "HavingCold"),
         sensations = replace(sensations, sensations=="Having fever", "HavingFever"),
         sensations = replace(sensations, sensations=="Having flu", "HavingFlu"),
         sensations = replace(sensations, sensations=="Having headache", "HavingHeadache"),
         sensations = replace(sensations, sensations=="Having stomach flu", "HavingStomachFlu"),
         sensations = replace(sensations, sensations=="Having toothache", "HavingToothache"),
         sensations = replace(sensations, sensations=="Longing for", "LongingFor"),
         sensations = replace(sensations, sensations=="Self-regulation", "SelfRegulation"),
         sensations = replace(sensations, sensations=="Sexual arousal", "SexualArousal"),
         sensations = replace(sensations, sensations=="Social exclusion", "SocialExclusion"),
         sensations = replace(sensations, sensations=="Social longing", "SocialLonging"))

# Create an innerjoin to match sensations between datasets
d3 <- suppressWarnings(inner_join(predDF, d2 %>% select(-DBSCAN_labels),
                                  by = c("feeling_labels" = "sensations")))

# Correlate our DBSCAN with theirs
corDBSCAN <- cor(d3$DBSCAN_labels, d3$DBSCAN_class)


# Sensation words happen to be arranged in the same row order in our case
# But still probably a good idea to check this
d4 <- read_csv(paste0(scriptPath, "/Data/Fig2_tSNE_coords.csv")) %>%
  rename(xcoord = Var1, ycoord = Var2)

# Make an innerjoin with d3 so that we can visually compare our DBSCAN with theirs
d5 <- inner_join(d4 %>% rename(sensation = Row),
                 d3 %>% select(sensation = feeling_labels,
                               OurDBSCAN = DBSCAN_labels,
```

```
                                OriginalDBSCAN = DBSCAN_class)) %>%
  mutate(OurDBSCAN = replace(OurDBSCAN, OurDBSCAN=="0", "Unclassified"),
         OurDBSCAN = replace(OurDBSCAN, OurDBSCAN=="1", "Negative Emotion"),
         OurDBSCAN = replace(OurDBSCAN, OurDBSCAN=="2", "Positive Emotion"),
         OurDBSCAN = replace(OurDBSCAN, OurDBSCAN=="3", "Homeostatic States"),
         OurDBSCAN = replace(OurDBSCAN, OurDBSCAN=="4", "Illnesses"),
         OurDBSCAN = replace(OurDBSCAN, OurDBSCAN=="5", "Cognitive Processes")) %>%
  mutate(OriginalDBSCAN = replace(OriginalDBSCAN, OriginalDBSCAN=="-1", "Unclassified"),
         OriginalDBSCAN = replace(OriginalDBSCAN, OriginalDBSCAN=="1", "Negative Emotion"),
         OriginalDBSCAN = replace(OriginalDBSCAN, OriginalDBSCAN=="2", "Positive Emotion"),
         OriginalDBSCAN = replace(OriginalDBSCAN, OriginalDBSCAN=="3", "Illnesses"),
         OriginalDBSCAN = replace(OriginalDBSCAN, OriginalDBSCAN=="4", "Cognitive Processes"),
         OriginalDBSCAN = replace(OriginalDBSCAN, OriginalDBSCAN=="5", "Homeostatic States")) %>%
  gather(whoseDBSCAN, clusterLabel, OurDBSCAN:OriginalDBSCAN) %>%
  mutate(whoseDBSCAN = factor(whoseDBSCAN, levels=c("OriginalDBSCAN", "OurDBSCAN"))) %>%
  mutate(clusterLabel = factor(clusterLabel, levels=c("Unclassified",
                                                      "Negative Emotion",
                                                      "Positive Emotion",
                                                      "Illnesses",
                                                      "Cognitive Processes",
                                                      "Homeostatic States")))

# Specify color labels
feeling_colors <- c("#636363", "#3182bd", "#de2d26", "#756bb1", "#31a354", "#e6550d")

Fig2_rep <- ggplot(d5, aes(x=xcoord, y=ycoord, group=clusterLabel, color=clusterLabel)) +
  geom_point(size = 2) +
  geom_label(label=d5$sensation, nudge_x=0.25, nudge_y=0.2,
             size=2, show.legend=FALSE) +
  scale_color_manual(name="DBSCAN Classification",
                     values=feeling_colors) +
  facet_grid(whoseDBSCAN ~ .) +
  xlab("t-SNE1") +
  ylab("t-SNE2") +
  pnas_theme +
  guides(color = guide_legend(override.aes = list(size=4))) +
  theme(legend.title = element_text(size = 12),
        legend.text = element_text(size = 10),
        axis.text=element_blank(),
        legend.position="bottom")
Fig2_rep
```