**REPLICATION METHODS**

*Obtaining data and importing into R*

The authors freely provided their data on a public repository, which we downloaded. However, the original data were encoded in Matlab’s proprietary format. In order to replicate the analysis using R, we wrote custom Matlab scripts to extract the relevant data and save them as CSV files that we could read using R. {JOEY: IN THE CONCLUSION, SAY A WORD ABOUT HOW OPEN SCIENCE ≠ REPLICABLE SCIENCE. HARD TO FOLLOW THEIR ANALYSIS PIPELINE, AND BASICALLY IMPOSSIBLE TO REPLICATE UNLESS YOU HAVE A WORKING COPY OF MATLAB, WHICH IS NOT FREE SOFTWARE}

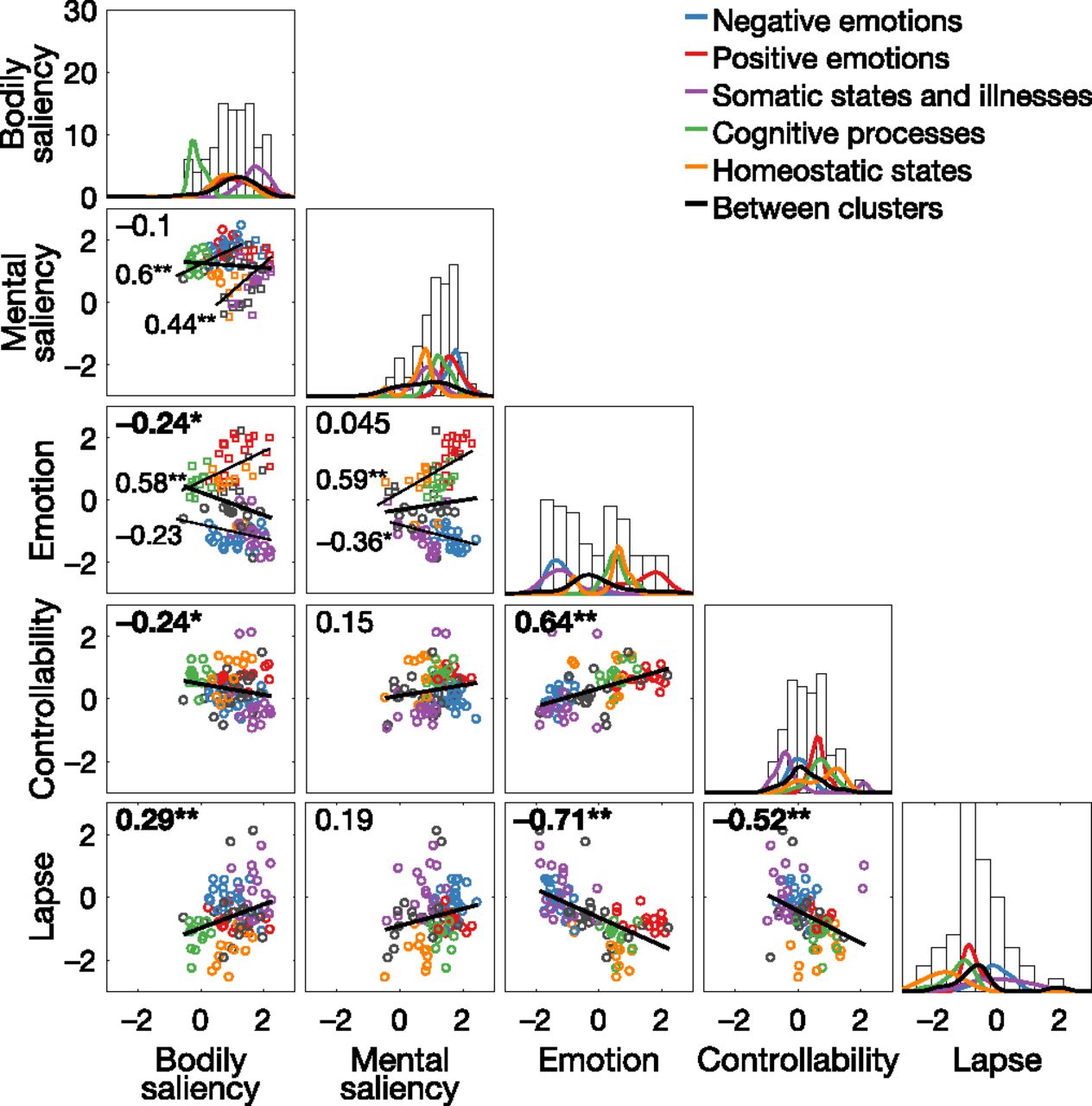
First, we extracted subjects’ mean ratings of how 100 common sensations varied along the following five dimensions: 1) the intensity of body sensations, 2) the salience of the mental experience, 3) the emotional valance (positive vs negative), 4) the perceived controllability of the sensation, and 5) how frequently the emotion occurs (operationalized as the time elapsed since the last experience of the sensation).

Second, we extracted results from the authors’ clustering analysis, which used machine learning techniques (i.e., DBSCAN, K-means, and hierarchical clustering) to categorize these 100 sensations into groups of related sensations. Third, we extracted the Cartesian coordinates for where each of the 100 sensations were placed in two-dimensional space using a dimensionality-reduction algorithm (t-SNE), which allowed us to visualize the results of the clustering analysis.

Finally, in order to replicate the clustering analysis using R’s implementation of the DBSCAN algorithm, we extracted the dissimilarity matrix for the 100 sensations. This dissimilarity matrix contains Euclidean distances quantifying subjects’ ratings for how similar (or equivalently, how dissimilar) any two sensations were judged to be.

*Goal 1: replicate Figure 1*

In the original paper, Figure 1 showed results from two analyses conducted in Experiment 1. First, for each of the 100 sensations, the figure plotted the Spearman correlation between subjects’ ratings for five author-defined dimensions (body saliency, mind saliency, emotional valence, controllability, and lapse). These scatterplots also included the ordinary least squares linear regression line drawn through the points. Each of the sensations visualized as a scatterpoint was color-coded according to how a machine learning clustering algorithm grouped related sensations in Experiment 2. Second, the figure showed histograms showing how 100 sensations’ mean ratings were distributed along each of the five author-defined dimensions.

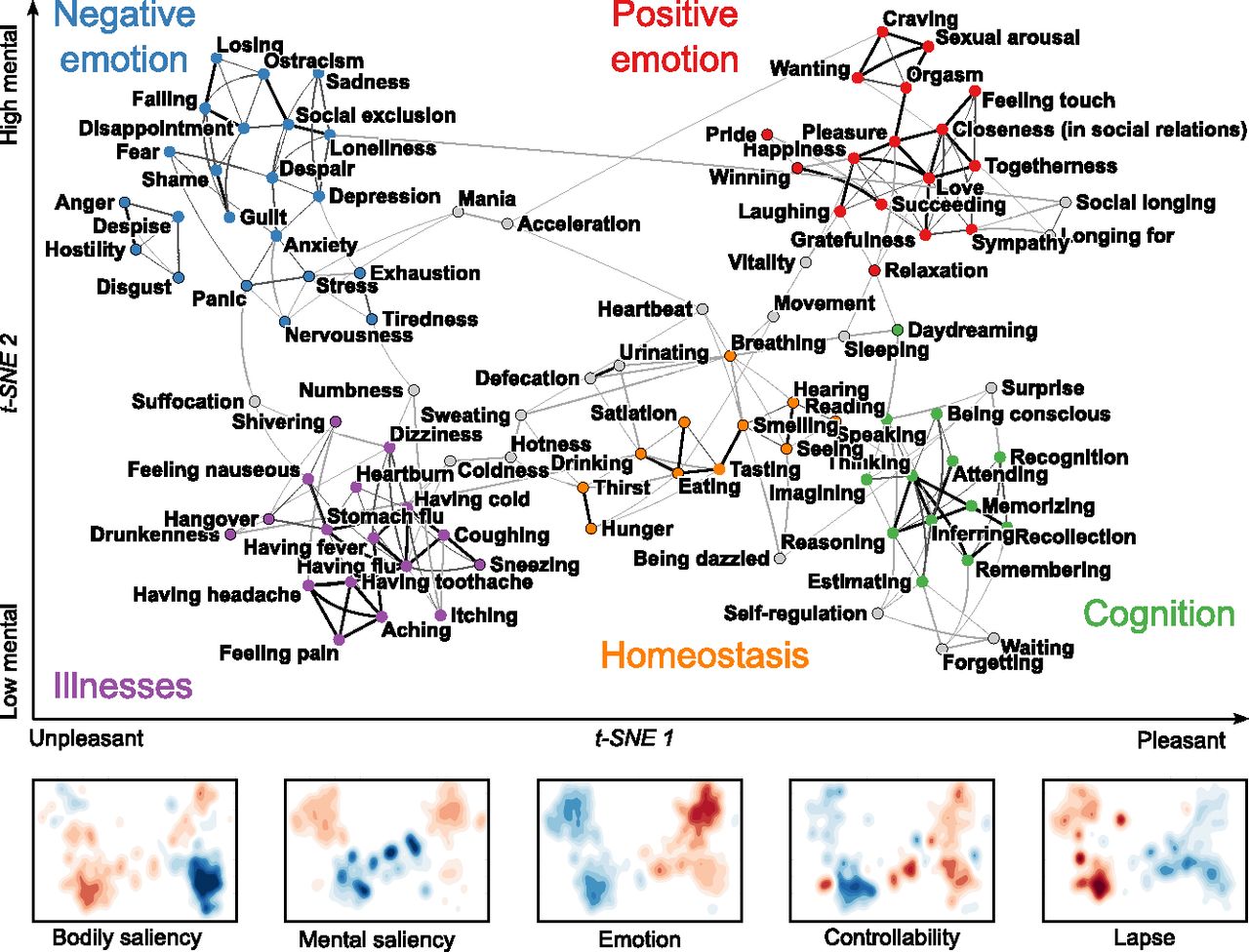


**Figure 1.** The figure that was reported in the original manuscript, provided as a visual reference.

In order to replicate this figure, we used the tidyverse library to tidy and visualize the data. We then used the cowplot library to “merge” all subplots into a single figure. Spearman correlations and p-values were calculated using the cor and cor.test functions in base R. For the scatterplots, we chose to report only the overall Spearman correlation, whereas the authors also reported Spearman correlations for each algorithm-defined cluster. Similarly, for the histograms, we chose to visualize only the overall distribution, whereas the authors also visualized distributions for each algorithm-defined cluster.

*Goal 2: replicate Figure 2*

In the original paper, Figure 2 visualized the results of the authors’ clustering analysis, which grouped the 100 sensations according to participants’ judgments of their similarity. They used a Matlab implementation of the DBSCAN algorithm to perform this analysis. In short, DBSCAN is an unsupervised machine learning algorithm that can identify irregularly-shaped clusters in density-based datasets, even in the presence of noise and outliers. Compared to other unsupervised machine learning approaches like K-means, DBSCAN has the following advantages: 1) it does not require the researcher to define a specific number of clusters to extract from the dataset; 2) it can find irregularly-shaped clusters, which is useful when the data is high-dimensional; and 3) it identifies outliers and is particularly robust to outliers.



**Figure 2.** The figure that was reported in the original manuscript, provided as a visual reference.

To replicate this analysis in R, we first extracted the Euclidean distances between all pairwise combinations of the 100 sensations, thereby creating a dissimilarity matrix. Next, we needed to define the two parameters used by DBSCAN. The first parameter, minimum points (MinPts), defines the minimum number of neighbors that should be contained in a neighborhood, with a radius defined by the second parameter, epsilon (eps). Therefore, eps defines the radius of a neighborhood surrounding specific points in density space. To summarize in simple English, our DBSCAN clustering method is a recursive process of finding clusters that containing at least MinPts members, then finding surrounding points that are within the radius eps and adding them to existing clusters. Points that do not fall into clusters using this method are therefore identified as outliers.

In order to search for the optimal value of eps, we used k-Nearest Neighbor Distances on the dissimilarity matrix, as implemented in the function dbscan::kNNdistplot. We used MinPts=5, the value used by the authors according to their Matlab code. This resulted in a plot of points sorted by distance, and their corresponding 5-Nearest Neighbors Distances. The optimal value of eps can then be found by searching for where there is a “knee” in the plot. Our visual analysis was concordant with the original analysis, which used eps=0.2565 according to the authors’ Matlab code. {JOEY: SAY A WORD IN THE CONCLUSION ABOUT HOW THESE PARAMETER VALUES WERE LEFT UNREPORTED IN THE ORIGINAL MANUSCRIPT + SUPPLEMENT. WE HAD TO DIG THROUGH THEIR MATLAB CODE TO FIND THIS INFO.}

With these DBSCAN parameter values, we used the fpc::dbscan function on our dissimilarity matrix to identify clusters of related sensations. {JOEY: SAY A WORD ABOUT HOW THE DBSCAN ALGORITHM IS IMPLEMENTED SLIGHTLY DIFFERENTLY IN R VS MATLAB.} To visualize our DBSCAN algorithm results, and to compare our results with the authors’ original clusters, we used the t-SNE algorithm (t-distributed stochastic neighbor embedding). This algorithm is a dimensionality reduction technique conceptually akin to a nonlinear PCA, and aims to represent dissimilarity in high-dimensional space by placing points closer together/further apart in two-dimensional space. Importantly, the use of t-SNE was not an analytic technique, but rather a convenient method for visualizing dissimilarity between sensations, and clusters of sensations. We loaded the results of the authors’ t-SNE analysis, which provided us with Cartesian coordinates for where each of the 100 sensations should be placed in two-dimensional space.

To recap in plain English, Figure 2 is a two-dimensional representation of a potentially high-dimensional “sensation space” and is therefore a map of sensations arranged by the t-SNE algorithm and grouped using the DBSCAN algorithm.

**EXTENSION METHODS**

*General strategy*

Using the DBSCAN algorithm, the authors drew strong conclusions about how sensations might be structured. However, this leaves open the question of whether alternative clustering methods would have led to similar interpretations of sensation structure. If not, this would imply that the conclusions drawn by the authors are highly contingent on the analytic method. Curiously, the authors’ public repository contained results from two other clustering techniques, which were not reported in either the manuscript or supplement. Therefore, we extended the published work by comparing the published clustering results against alternative clustering methods.

*K-means clustering*

One of the simplest and most popular clustering techniques in unsupervised machine learning is K-means clustering. The intuition behind this technique is conceptually related to ordinary least-squares regression: the researcher specifies that they want to extract K=k clusters from the data, and the algorithm assigns data points to clusters by minimizing the within-cluster sum of squares until an “optimal” clustering is found that minimizes the sum of squares. Here, the authors specified that they wanted k=5 clusters, to match the number of clusters obtained by DBSCAN. To compare the results of the K-means clusters against the DBSCAN clusters, we mapped sensations to the same Cartesian locations in t-SNE coordinate space, then color-coded them according to their clusters. We assigned qualitative labels to these clusters based on our best guess of what real-world structure found by the algorithm.

*Hierarchical clustering*

Another method for clustering dissimilarity data is hierarchical clustering, an algorithm whose goal is to create a dendrogram reflecting the hierarchical structure of multilevel data. Therefore, in a three-level dendrogram, two distinct clusters at the third level would be subcomponents of the same cluster at the second level. The authors used an agglomerative approach (i.e., a bottom-up approach), in which each datapoint is first treated as its own cluster. The algorithm then recursively merges the two clusters that are closest together until all points have been merged into a single cluster (i.e., the first-level parent cluster). After implementing this algorithm, the authors identified the Nth level where the minimum number of points within the smallest cluster was equal to the smallest number of points within the DBSCAN clusters. Since they set the DBSCAN parameter MinPts=5, this ended up being a minimum of five points within the smallest cluster. As we did with K-means clustering, we mapped the sensations using t-SNE coordinates, then color-coded according to the identified clusters. Once again, we assigned qualitative labels to these clusters based on our best guesses about the underlying structure.