Master's Thesis

Interactive Visualization of Nonlinear Dimensionality Reduction Methods for Complex Data

Department of Statistics
Ludwig-Maximilians-Universität München

Judith Jennert

Munich, January 23rd, 2024

Submitted in partial fulfillment of the requirements for the degree of M. Sc.
Supervised by Dr. Fabian Scheipl

Abstract

When working with complex data like functional or image data the challenge of effective visualization arises. Assuming the existence of an underlying manifold, it is possible to approximate it with linear or nonlinear dimensionality reduction methods such as MDS, t-SNE or UMAP. The resulting embeddings then allow for scatterplot visualizations. The aim of this thesis is to provide tools to facilitate these visualizations as well as connect them intuitively and hassle-free to the actual data. Interactive apps tailored to both functional and image data make comprehensive and intuitive visualization of embeddings quickly accessible. They let users interact with individual or selected groups of data points, visualize the represented data, and categorize them by additional information. They also allow for comparison of different embedding dimensions and methods. The application of these apps on real-life data to compare embedding methods suggests that valuable insights can be gained, particularly when working with functional data.

Contents

1 Introduction

Complex data is all around us and its analysis as relevant as it ever was. With the increasing availability of data and computational capabilities, there is considerable interest in methods that enable access to the information embedded in the data. These datasets are often high-dimensional and require specific techniques for meaningful analysis.
Visualizing such data in its entirety, not just as single observations, is as challenging as it is necessary, particularly during the exploration process. Non-linear dimensionality reduction methods, such as MDS, t-SNE and UMAP (see Cox and Cox, 2008, van der Maaten and Hinton, 2008, McInnes et al., 2018), provide techniques to make visualization of high-dimensional data feasible.
Embedding data into lower dimensions can also serve as preparation for downstream tasks such as outlier detection or clustering methods (see e.g. Herrmann and Scheipl, 2021). They should, however, be applied with care, and visual inspection is a valuable screening tool for unsupervised methods.
It is therefore convenient to have a user-oriented tool available, which assists in the visualization of complex data and simplifies the process of exploratory analysis. Such a tool proves helpful at various stages of data analysis whenever the visualization of both the embeddings and the complex data observations themselves are required. This thesis presents two R Shiny apps that aim to provide such a tool. One is tailored towards functional data, the other for image data. By making the visualization process interactive, they facilitate the connection of embedding point and original data, as well as the visualization of further information. Known classes, scoring calculated on the data or other additional variables can be incorporated interactively. Various visualization designs assist in the intuitive comprehension of embedding spaces with more than two dimensions.
To demonstrate the usefulness both of the described dimensionality reduction methods and the apps, selected results from exploratory analyses conducted on real-life biomedical data, as well as toy data examples, are included.
A short overview over the structure of this thesis is as follows: section 2 introduces and defines the concept of complex data and the specific data types used in this work. It also describes common challenges arising when working with complex data. Then, section 3 provides the theoretical background for tackling those challenges. The dimensionality reduction methods used in the thesis are described in section 4 and compared in section 5. The practical part of this thesis consists of section 6 and section 7. The former gives a detailed description of the apps developed, the latter provides exemplary analyses of

complex data using the described methods and apps. A discussion of the results and concluding remarks can be found in section 8.

2 Complex data

We are constantly surrounded by complex information that is treated as high-dimensional data. The music we listen to, the images we scroll through, all of it is stored as and can be represented as high-dimensional data. Complex Data can be described as data that goes beyond the traditional statistical collection of data, where each observation is described by a limited number of parameters. Complex data usually consists of many more 'parameters', which may be connected by various underlying structures. In other words, we are moving within "high dimensional spaces, which arise from techniques that approximate [...] complex data [...] with long 'feature' vectors" (Beyer et al., 1999, p.217).
This thesis focuses on two types of data typically described as complex and high-dimensional: functional data and image data. Functional data (FD) is theoretically infinite-dimensional with strong structure. Firstly, "while FD are infinite-dimensional in theory, they are high-dimensional in practice: functional observations are usually recorded on fine and dense grids of argument values. Secondly, the dominant drivers of the differences among functional observations are often comparatively low-dimensional, so that just a few modes of variation capture most of the structured variability in the data" (Herrmann and Scheipl, 2021, p.972). In image data represented by pixel values, not only the location and value of the individual pixels are relevant, but for example their relative location to each other. It is less relevant if a certain pixel is black, and more relevant if the ones around it are of a similar color or not. Like for most complex data types, this complex high-dimensional data especially calls for specifically tailored methods.
Lee and Verleysen (2007) comment on possible perspectives to be taken when working with high-dimensional data. On the one hand, there is the practical perspective focused on tangible results. Among typical objectives in unsupervised methods for dimensionality reduction are visualization, clustering, outlier detection, feature extraction, representation learning, and manifold learning. "On the other hand, theoretical motivations relate to the study of high-dimensional spaces and distributions. Their properties prove unexpected and completely differ from what is usually observed in low-dimensional spaces" (Lee and Verleysen, 2007, p.1).
In practice, the threshold for data being called high-dimensional is usually if $p > n$, p being the number of parameters and n the number of observations. This is not a hard

and fast rule though, and other factors may also play a role in nomenclature, e.g. if a dataset is seen as multivariate or high-dimensional. Beyer et al. (1999) for example define just 10 - 15 dimensions as high-dimensional.
There is an important distinction to be made, which is whether the dimensionality is intrinsic or merely observational. The relevance is discussed in more detail in section 3.
There are many visualization tools which help to intuitively understand (certain aspects of) a dataset. These usually are not suitable for high-dimensional data as is, which contributes to the difficulty of exploring and gaining an understanding of complex datasets. Dimensionality reduction is a way of tackling this challenge, as various visualization (and summarization) methods become viable on lower dimensionality.

2.1 The curse of dimensionality

Generally, there are "issues of efficiency and efficacy" (Zimek et al., 2012, p.364) that arise when working with high-dimensional data. One of the main challenges in high-dimensional data is the so-called 'curse of dimensionality'. This symbolic term was first introduced by Bellman (1957) and has since become commonly used in a number of fields. It encompasses a number of unexpected and un-intuitive phenomena that can be observed as the dimensionality of the analyzed data increases. Specifically, "more concrete aspects being the so-called 'distance concentration effect' [or] the presence of irrelevant attributes concealing relevant information" (Zimek et al., 2012, p.363). While Zimek et al. (2012) focus on the relevance of the curse of dimensionality (CoD) on outlier detection, their descriptions of the effects are relevant for the search for sensible embeddings in general as well. In the following, I summarize some of the problems mentioned.

- **Distance concentration effect**

  The concentration effect describes how pair-wise distances loose a lot of significance in high dimensions because their relative differences approach zero in sparse high-dimensional space (see Zimek et al., 2012, p.364). More precisely, the variance of distances shrinks as the number of dimensions increases. Given some broad conditions (broader than the data being iid[1]), the proportional difference between the distance of the closest neighbor to the distance of the farthest neighbor decreases (see Beyer et al., 1999).

  This makes it hard to differentiate similar data points from dissimilar ones. As the embedding methods described in this thesis all depend on distance or at least

[1]For a list of abbreviations see Table 1

dissimilarity measures, this is a relevant factor. Zimek et al. (2012) give concrete examples of the effect in action, using iid data of various distributions and show the average distance converging with a shrinking variance. In this setting, the effects can be attributed to the central limit theorem (see Zimek et al., 2012, p.365).

- **Noise**

  If there are a lot of un-informative dimensions, they will add a lot of noise and necessarily hide the relevant features.

- **Sparsity of data**

  The more variables there are, the more data is necessary, to grasp the distribution over all the dimensions. This is an exponentially increasing relationship.

- **Hubness**

  In high dimensions, there appear to be some data points that are many other's nearest neighbor, while many are not the nearest neighbor of any other point.

- **Combinatorial explosion**

  The more dimensions there are, the more likely within-distribution outliers are to appear in at least one dimension. Increasing the number of dimensions without specific reasoning can be seen as a form of overfitting (see Zimek et al., 2012). Counteracting this via methods used in lower-dimensional spaces (e.g. grid-based approaches) is not feasible for a higher number of dimensions, as the space having to be covered grows exponentially.

The CoD does come with an important asterisk. The assumption is that the CoD can be mitigated or will not even come into effect under certain conditions. For example, if each of the many dimensions adds information, it actually becomes easier to extract information, rather than harder. "We see here that this kind of outlierness [namely off-manifold outliers] becomes not less but ever more prominent when increasing the dimensionality (as long as the dimensions add information)" (Zimek et al., 2012, p.366). In other words, outliers that are drawn from a different distribution in each of the dimensions may reverse the effect of the CoD.
Also, if the intrinsic dimensionality of the data is much smaller than the observed one and this underlying structure is approximated, the curse can be mitigated. In the context of machine learning this is called 'feature extraction'. This is often the case in functional

data, as the functions may be sufficiently described by a specific set of parameters and some noise. The assumption is, therefore, that the low-dimensional functions are embedded in a high-dimensional observation space (see Herrmann and Scheipl, 2021). The theoretical background is further explained in section 3.
If a suitable embedding is found, data can be handled more efficiently. Relevant information should no longer be concealed, although a loss of information is usually unavoidable.

2.2 Functional data

Among the different types of high-dimensional data, this thesis mainly concentrates on functional data. According to Ramsay (1982), "[a] datum is often a continuous function x(t) of a variable such as time observed over some interval. One or more such functions are observed for each subject or unit of observation". As hinted at in that statement, measurements do not have to be made over time, one alternative would be spatial location. In essence, it has to constitute a continuum. Herrmann and Scheipl (2021) describe functional data as "complex and information-rich units of observations".
Related definitions include multivariate and longitudinal data, at times the classification comes down to perspective. Müller (2005, p.224) writes about "infinite-dimensional data, such as curves, shapes and images", using a broad definition of functional data.
Functional data analysis (FDA) is the statistical methodology comprised of various definitions, fields of research, tools, and techniques for studying and working with data defined as functional. "The basic philosophy of functional data analysis is to think of observed data functions as single entities, rather than merely a sequence of individual observations" (Ramsay and Silverman, 2005, p.38). Among the goals of FDA are finding representations of the data suitable for further analysis, highlighting characteristics, and identifying and analyzing the sources of pattern and variation found in the data (see Ramsay and Silverman, 2005, p.9).
In practice, the data are observed discretely, e.g. as measurements y_i at points t_i, $i = 1, ..., N$; $i \in \mathbb{N}$. The recorded dimensionality is therefore given by N, the number of measurements per observation.
Among the various challenges posed by this type of data are the dimensionality, observation noise, as well as the data containing shape, translation and phase variation (see e.g. Herrmann and Scheipl, 2021, Marron et al., 2015).
A detailed description of the functional data used in this thesis for the app and for analysis can be found in subsubsection 7.1.1.

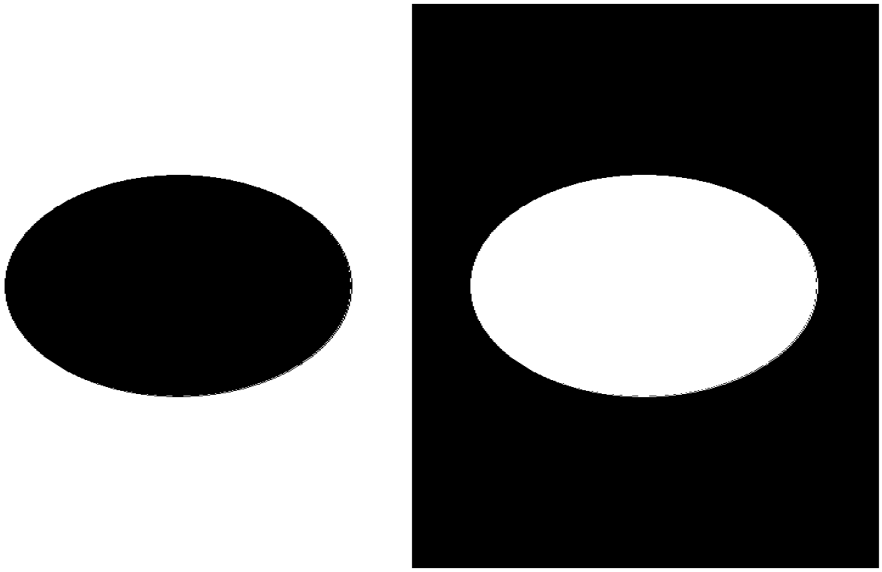

Figure 1: Two pictures of an ellipse, with maximum distance based on pure pixel values

2.3 Image data

To capture an image's essence in its representation can be rather complex, as there is often a lot of variability even within images of, for example, the same object. There can be variation in color, rotation, texture, scale and even which parts of the subject are included (see Herrmann et al., 2022, p.16).
There are several ways in which images can be represented. One option is to see each pixel as a dimension and represent each image as a vector of pixel color values. This often does not lead to satisfying results, as many aspects and characteristics of the image are not taken into account. Changes of position of objects in an image for example are not being recognized. Imagining black and white pictures of hand drawn numbers, similar numbers drawn in different locations will yield far distances calculated on such pixel vectors. If in one picture the zero is drawn completely in the left half of the image, and in the other it is completely in the right half, then none of the black pixels would overlap. The calculated distance would be greater than that of a number eight drawn in the same position as the number zero. Such a distance calculation would therefore mainly encode information about the location of the drawn digit, and less about the maybe more interesting aspects, namely which digit it encodes. Another example can be seen in Figure 1. The two pictures closely resemble each other, yet have the maximum distance from each other in grey scale pixel values. Any other picture, also represented by black and white, would be closer in distance to either of those two images than they are to each other.

A regularly used technique to find potentially more relevant dimensions to represent the images is by employing a model for feature extraction. The resulting representation embedding consists of many dimensions (sometimes more than the original picture values), where each of the dimensions ideally carries relevant information. The representation should therefore more closely resemble the intrinsic dimensionality of the image data point.
Another challenge, apart from aptly capturing an image's characteristics, is that image datasets usually come with large data volumes. Huge amounts of digital image collections are available, from private use to medical imaging. With technical advancements the resolution of these images also increases.
Comprehensively visualizing an image dataset presents a challenge, as "at most a few observations can be visualized and perceived simultaneously" (Herrmann et al., 2022, p.16). This makes well-suited low-dimensional embeddings of image datasets valuable, as they "provide easily accessible visualizations of the data" (ibid).
A detailed description of the image data used in this thesis can be found in subsection 7.2.

3 The manifold assumption

Seeing as nonlinear dimensionality reduction methods are also referred to as 'manifold learning', it is unsurprising that they operate on the assumption that the observed data lies on a manifold which can be inferred and represented in lower-dimensional space. A manifold is defined as a topological space which locally resembles Euclidean space at any given point on the manifold. Furthermore, the assumption can be made that the intrinsic dimensionality of the data is lower than the observed one. Using functional data as an example, a theoretically infinite-dimensional function is finite-dimensional when described by its parameters. This idea of lower intrinsic dimensionality can be generalized to a lot of observationally high-dimensional data, like image data. Meaningful image representations contain structure (like edges, shapes, information about components) rather than a potentially infinitely fine grid of color values. In the following I concentrate on functional data as a stand-in for high-dimensional data in general.
Following the framework developed by Herrmann et al. (2022), a high-dimensional functional manifold can be mathematically described as follows.
Let $\mathcal{F}$ be the observation space of dimension D. The parameters $\theta_i \in \Theta$, with $\Theta \subset \mathbb{R}^d$, shall be realizations of the probability distribution P, so $\theta_i \sim P$. Let $\mathcal{Y} \subset \mathbb{R}^{d'}$ be an embedding space. Then define ϕ and e as mappings, so that

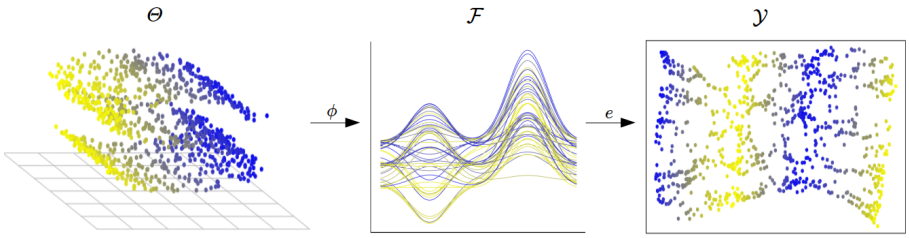

Figure 2: Functional data from a manifold-learning perspective. Image source: Herrmann and Scheipl (2021).

$$\Theta \xrightarrow{\phi} \mathcal{M}_{\mathcal{F}} \xrightarrow{e} \mathcal{Y}.$$

with $\mathcal{M}_{\mathcal{F}} \subset \mathcal{F}$ being a manifold in the observation space, defined by Θ, P, and ϕ. The map ϕ preserves lengths and angles, given the metric, i.e. it is isometric (see Herrmann et al., 2022, p.5). The assumption is then, that a low-dimensional representation in $\mathcal{Y}$ can be found that both approximates the metric structure of $\mathcal{M}_{\mathcal{F}}$ and captures as much of the parameter space Θ structure as possible. Such an embedding would therefore ideally be $y_i = e(\phi(\theta_i)) \in \mathcal{Y}$ (see Herrmann et al., 2022, p.5).
Among other embedding methods, both t-SNE and UMAP rely strongly on the manifold assumption. In general, for an embedding to be able to represent the true underlying structure, such a manifold structure has to be assumed to exist.

3.1 Significance for practical application

If the underlying manifold of the data can be successfully approximated, it is therefore possible to make the geometric structure of the dataset visible with the help of manifold learning techniques. This can offer valuable insights into the data landscape in exploratory visual analysis and lay important groundwork for further inspection of the data. If the manifold is well approximated, embedding techniques can, for example, be used for clustering tasks as well as outlier detection. While these questions will initially be investigated through visual inspection of the embeddings and the data, it is possible to use the embedded data for automated clustering or outlier detection techniques. It is then vital to make sure the chosen embedding method and setting is suitable for the specific task. Herrmann and Scheipl (2021) and Herrmann et al. (2022) describe how dimensionality reduction techniques can be combined with outlier detection methods, such as using LOF-scores on embeddings.
Definitions for outlyingness in functional data tend to be fuzzy. Given the geometric

framework described above, it is possible to define a clear classification of outlyingness for functional and other high-dimensional data. Herrmann and Scheipl (2021) introduce such a well defined, theoretically substantiated framework. "Unlike these single-manifold settings, our conceptualization of outlier detection is based on two functional manifolds" (Herrmann and Scheipl, 2021, p.974). If we observe a dataset $X = \{x_1(t), ..., x_n(t)\}$ with n functional data points, we assume the existence of two functional manifolds these observations are coming from, one representing the majority data-generating process ($\mathcal{M}_c$), and the other containing data not generated by that process ($\mathcal{M}_a$). Note here, that the data contained in the latter does not have to be similar to each other at all. Herrmann and Scheipl (2021) use the phrasing "'common' process generating the bulk of observations on ($\mathcal{M}_c$), and an 'anomalous' process defining structurally different observations on ($\mathcal{M}_a$)." Given this definition, outliers within the data can be split into two types: off-manifold outliers and on-manifold outliers. If a functional observation $x_i(t) \in \mathcal{M}_a$ and $x_i(t) \notin \mathcal{M}_c$, it is an off-manifold outlier. An outlying functional observation $x_i(t) \in \mathcal{M}_c$ with $\theta_i \notin \Omega^*_\alpha$ is an on-manifold outlier. Ω^*_α is defined by P. Off-manifold outliers are therefore *structural outliers*, on-manifold outliers are *distributional outliers* (see Herrmann et al., 2022, p.6).

"We argue that such a perspective both clarifies and generalizes the concept of functional outliers, without the need for any strong assumptions or prior knowledge about the underlying data-generating process or its outliers" (Herrmann and Scheipl, 2021, p.971f.).

4 Embedding methods

Dimensionality reduction techniques are used for many reasons. The difficulties that come with high dimensionality can be circumvented, when embedding the data into low-dimensional spaces.In this thesis, three of the most commonly used methods are described and used for example analyses in section 7, namely MDS, t-SNE and UMAP.

4.1 MDS

When dealing with high-dimensional data, like other embedding methods, multidimensional scaling (MDS) makes it possible to visualize the data in low(er)-dimensional space. This opens up the possibility of finding similarity structures or even clusters present in the data, becoming aware of outliers, and in general ascertaining which data points are comparatively similar to each other, and which are not. In other words, it can be a

valuable tool in exploratory data analysis, for visual qualitative inspection, but also for further quantitative analysis. Depending on the question at hand, more importance may be given to local or global similarity structures, the focus of which informs the choice of embedding method. As MDS ideally preserves the distance structure, both global and local structures are kept if a good fit can be found. Visualizing these relationships between data points and their relative distances or dissimilarities to each other can be helpful both for gaining insights into an unknown dataset, as well as for getting a better understanding of a more familiar one.

MDS is a technique used to map a distance (or dissimilarity) structure of observations to a k-dimensional space, where $k < p$ (p = observed dimensions). First versions appearing in the 1930s and popularized by Torgerson (1952), it is a well known and widely used technique. The literature usually differentiates between metric and non-metric scaling (see Cox and Cox, 2008) (although the distinctions and the naming are not used uniformly). Whereas the classical, metric scaling calculates the lower-dimensional values analytically, using eigenvectors and -values of the dissimilarity matrix, the non-metric version optimizes over a loss function called 'stress' (or sometimes 'strain'). The stress function minimizes the differences between the original pairwise distances and the pairwise distances in the new, lower-dimensional space.

The term 'metric MDS' in particular is used variably in literature, sometimes referring to the classical approach, sometimes generalizing it over metric, non-Euclidean distances and optimizing over the stress function as well, being differentiated from non-metric MDS by the latter using only ordinal dissimilarities.

The data analysis in section 7 was performed using classical (metric) MDS as described in Mardia (1978) and Cox and Cox (2008), which use classical MDS based on Torgerson (1952).

4.1.1 Computation

In the following, I will describe the computation of the classical MDS embedding. Let X be a matrix containing observations of a high-dimensional dataset, each observation existing in $\mathbb{R}^m, m \in \mathbb{N}$. Let D be a symmetrical $n \times n$ matrix containing the pairwise non-negative distances (or continuous parametric monotonic transformations of dissimilarities, to get them to a "distance-like form" (Cox and Cox, 2008, p.319)) for each of the n data points, with the diagonal x_{ii} being 0.

To get the coordinates in the lower-dimensional space with k dimensions, start with a symmetrical Matrix A, the non-diagonal entries being $-\frac{d_{ij}^2}{2}$. From there, find a new matrix

$B = HAH$, with $H = \mathbb{I} - n^{-1}\mathbf{1}_n\mathbf{1}_n^T$ being the centering matrix, where $\mathbf{1}_n$ is a vector of ones, and $\mathbb{I}$ the n-dimensional identity matrix. Through spectral decomposition of B, find its eigenvalues and the corresponding eigenvectors, so that $B = V\Lambda V^T$, with Λ containing the eigenvalues on its diagonal, and V column-wise containing the corresponding eigenvectors. Keep only the first k eigenvalues and corresponding eigenvectors, resulting in Λ_1 and V_1. Should there be non-positive eigenvalues, those get dropped as well, even if this results in less than k dimensions being returned.
The rows of the new matrix $X^* = (V_1\Lambda_1)^{1/2}$ contain the coordinates for the observations in the k-dimensional embedding space.
For smaller ks, the resulting coordinates can be visualized in scatterplots. If these dimensions carry enough information, this allows for intuitive visualization of the data structure. The choice of k is an important hyperparameter, as it can result in underfitting if too small or in overfitting if too large, also making it harder to intuitively visualize.
As a metric for the quality of the embedding in representing the high-dimensional structure, I used the metric Goodness of Fit (GOF), calculated as follows:

$$GOF = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n-1} |\lambda_i|}$$

.

In case the number of the positive eigenvalues is less than the number of new dimensions k, the GOF is calculated slightly differently:

$$GOF = \frac{\sum_{i=1}^{k} \lambda_i}{\sum(positive\ eigenvalues)}$$

.

Solutions obtained by a MDS embedding are not unique. Adding any constant to all coordinates for example, or any arbitrary rotation, results in an equivalent solution, as distances are being preserved in the lower-dimensional Euclidean space.
When using classical MDS on the Euclidean L2 distance, the results are identical to performing PCA (Principal Component Analysis).

4.1.2 Practical application

One of the questions arising when using MDS embeddings is to determine the dimensionality needed to capture enough of the data structure to still be meaningful. Ideally

one would want to be able to represent the intrinsic dimensionality of the data, if it is actually relatively low-dimensional (see section 3). One option is to visually inspect the GOF-values for different numbers of dimensions (or scree plot of the stress, when using non-metric versions), and look for a sharp drop or 'elbow' in the plot. However, it is often the case that "the pattern of change of data-fit across dimensionality, rather than containing an elbow, is often better characterized as one of smooth and gradual decline" (Lee, 2001, p.152) or increase. This describes my experience while embedding mouse hearing curves (described in subsubsection 7.1.1) well, as can be seen in Figure 3.
MDS uses a distance matrix as a basis for an embedding rather than the actual data. This means on the one hand, that the choice of distance function used is an important design decision when using MDS. This can be advantageous when the Euclidean distance does not capture the underlying distance structure well. It is also valuable in cases when no distance can be calculated, e.g. when there is only ordinal information and therefore only a dissimilarity matrix is available.
Among the most commonly used distance functions are the Euclidean distance, the city block metric (or more generally, the Minkowski metric with a chosen power), and the Canberra metric. There are various distance functions suited for specific data types, e.g. the Levenshtein distance being used for linguistic strings or biological strings like DNA.
The more common distances mentioned above are calculated as follows (see Cox and Cox, 2008, table 3.1):

The Minkowski metric is defined as

$$\delta_{rs} = \{\sum_i w_i |x_{ri} - x_{si}|^\lambda\}^{1/\lambda} \quad \lambda \geq 1,$$

with the special case of the Euclidean distance (power of 2)

$$\delta_{rs} = \{\sum_i w_i |x_{ri} - x_{si}|^2\}^{1/2}$$

and the city block metric (power of 1)

$$\delta_{rs} = \sum_i |x_{ri} - x_{si}|.$$

The Canberra metric

$$\delta_{rs} = \sum_i \frac{|x_{ri} - x_{si}|}{x_{ri} + x_{si}}$$

is a weighted version of the city block metric.
Herrmann and Scheipl (2021) describe the benefits of using MDS in combination with outlier detection methods such as using LOF-scores on the embedding coordinates. This, too, requires the choice of a fitting distance measure.

4.2 t-SNE

The nonlinear dimensionality reduction method called t-distributed stochastic neighborhood embedding (t-SNE) (van der Maaten and Hinton, 2008) is a variation of the method SNE (Hinton and Roweis, 2002). It quickly became a widely used method for non-linear dimensionality reduction and continues to be a standard implementation choice.
Other methods like MDS tend to prioritize the preservation of a global structure of the data, meaning bigger distances between data points are preserved in the results, rather than smaller distances between points similar to each other. These relationships between similar points are what t-SNE is after, as it shifts the focus on the local structure of the data. t-SNE takes pairwise affinities of data points and models them using joint Gaussian distributions in the original high-dimensional space and joint Student t-distributions in the lower-dimensional space, minimizing the divergence between the distributions. The heavier tails of the Student t-distribution have the effect that greater distances are stretched more in the target dimensions, so that local neighborhood distances are kept while mapping into lower dimensions. This favors clustering and clear separation of said clusters. On the other hand, absolute distances between non-neighbors in the embedding space are not meaningful.
The final coordinates depend on a variety of chosen hyperparameters, implementation choices, as well as randomization like the initializing coordinates. Therefore, results can vary from implementation to implementation even when the settings are kept the same. Standard practices include strategies for improving results, e.g. performing t-SNE on an initial PCA embedding. As the loss function is not convex, there is no guarantee a global optimum will be found.
t-SNE works on the assumption that the data lies on a manifold and therefore has lower intrinsic dimensionality (van der Maaten and Hinton, 2008, p.2580).
Among problems that may arise in practice, one is that by prioritizing local structure significant global structure may be lost. Also, because t-SNE is designed to find cluster representations, structures might be shown in the resulting embedding space, that do not actually exist in the original data. Because of the optimization process being constricted

by a maximum number of iterations and able to be fooled by local optima, there is no guarantee the found representation is ideal, the best possible, or even fitting. Because of the focus on local structure and small distances and neighborhoods, it does not lend itself that well to outlier representation either, as outliers are more likely to be clustered with neighboring points.
Nonetheless, t-SNE enables intuitive visualization of big and high-dimensional datasets and presents a flexible and practical option for exploratory analysis.

4.2.1 Computation

To find representative coordinates in low-dimensional space, t-SNE minimizes the divergence between the joint probabilities p_{ij} and q_{ij} in the high-dimensional space and the low-dimensional space respectively (see van der Maaten and Hinton, 2008)[1].
First, the data is normalized and distances D_{ij} are calculated. Then a Gaussian distribution $\mathcal{N}(x_i, \sigma^2)$ is centered on each data point x_i.
The standard deviation σ_i shall now be determined. It will vary depending on the density of the dataset at x_i, so as to represent neighborhoods, not absolute distances. The probability $p_{j|i}$, that data point x_i would pick data point x_j as its neighbor in the observed high-dimensional space, is set as

$$p_{j|i} = \frac{\exp(-\|D_{ij}\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|D_{ij}\|^2/2\sigma_i^2)}. \tag{1}$$

Now, we can find σ_i, such that the following equation is satisfied for the user-specified perplexity

$$perplexity = 2^{-\sum_j (p_{j|i} log_2 p_{j|i})}. \tag{2}$$

The resulting probabilities are then symmetrized via

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n},$$

with n being the number of observations.

[1]This probability-based approach can also be interpreted as a graph-based method, where the pairwise probabilities correspond to graph edges. That perspective is taken in the UMAP method (see McInnes et al., 2018)

The corresponding probability q_{ij} in the lower-dimensional space is calculated using a Student t-distribution with 1 degree of freedom (also called Cauchy-distribution)

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}. \tag{3}$$

As a method, t-SNE seeks to find embedding coordinates y_i in the lower-dimensional space, such that the divergence between p_{ij} and q_{ij} is minimal. Initial coordinates in the lower-dimensional space are sampled from a normal distribution centered around 0 and early exaggeration is used during optimization to facilitate clustering.
The divergence between the two distributions is minimized using the Kullback-Leibler divergence C, which results in the gradient

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 - \|y_i - y_j\|^2)^{-1}. \tag{4}$$

As the minimization using simple gradient descent has $O(n^2)$ complexity, a Barnes-Hut method is used (see van der Maaten, 2014) to make calculation feasible, bringing complexity down to $O(nlog(n))$.
To summarize, the following (simplified) default algorithm is employed to find the lower-dimensional t-SNE embedding:

1. Compute high-dimensional p_{ij} with σ_i determined by the *perplexity* (equation (1) and (2))
2. Set initial coordinates, e.g. using a PCA embedding
3. Repeat for the number of iterations:

   compute low-dimensional q_{ij} (equation (3)), then compute the gradient $\frac{\partial C}{\partial Y}$ (equation (4)) and set

   $$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)}),$$

   where η is the learning rate, and $\alpha(t)$ is the momentum

4.2.2 Practical application

t-SNE offers a number of hyperparameters and design decisions to adjust to specific data and research interests. Selecting the appropriate settings is less tricky, when the dataset and the domain are well known. Without further information, it is a challenge, as "hyperparameter tuning for unsupervised methods remains an unsolved problem" (Herrmann and Scheipl, 2021, p.988). One of the main decisions is the perplexity, which determines the size of the neighborhood that is to be taken into account. The bigger the perplexity is set, the more closely t-SNE will try to preserve global structure. While the authors claim that the "performance [...] is fairly robust to changes in the perplexity, and typical values are between 5 and 50" (van der Maaten and Hinton, 2008, p.2582), others see t-SNE as sensitive to the perplexity parameter (see e.g. Wang et al., 2021, p.2). They also point towards the propensity of t-SNE to "create spurious clusters" (ibid).
This is a non-exhaustive list of decisions, which can be set or changed by the user:
Perplexity, momentum α, learning rate η, θ for a speed vs. accuracy trade-off, maximum steps for iteration, and setting initial coordinates.
Initial coordinates in the lower-dimensional space can be chosen by the user instead of using a PCA embedding, or can be randomly set by sampling from $\mathcal{N}(0, 10^{-4}I)$. Coordinates found by dimensionality reduction methods preserving global structure lend themselves to be sensible initial coordinates to be further fitted via t-SNE. In that case, early exaggeration is not used, as the coordinates are assumed to already be descriptive.

4.3 UMAP

First introduced by McInnes et al. (2018), Uniform Manifold Approximation and Projection (UMAP) is one of the more popular embedding methods for complex data at the time of writing. It combines many of the advantages of t-SNE with (usually) faster computational times and offers flexibility when it comes to data types as well as tasks.
While t-SNE is bounded to embedding into up to three dimensions only by computational complexity, UMAP can embed into an arbitrary number of dimensions. Not only does this allow for a potentially truer depiction of the underlying manifold, it also makes it possible to use UMAP as a feature extraction task for machine learning. Where t-SNE mainly preserves local structure, UMAP also attempts to keep more of the global structure, such as distances between clusters or non-neighbors, although the focus is still kept on the local neighborhoods. As the computations of t-SNE and UMAP embeddings follow a similar structure, and can perform similarly under specific hyperparameter settings (see

McInnes et al., 2018, p.49), it makes sense to compare the two methods and see UMAP as belonging to the same family of embedding methods as t-SNE. The authors state that the spectral layout based on Laplacian Eigenmaps plays a significant role in UMAP generally preserving more of the global structure than t-SNE. When initializing t-SNE the same way, the biggest difference in this respect therefore lies in the different objective functions (see McInnes et al., 2018, p.49).
The UMAP algorithm design is based on arguments derived via mathematical theory rather than experimental. McInnes et al. (2018) draw on topology, graph theory, and category theory to lay the theoretical mathematical foundations and motivations for the methods employed by UMAP.
UMAP's focus is on topological information, which it wants to retain. It assumes uniformly distributed data on the manifold, or in other words, that the manifold is locally connected and there are no completely isolated points. (This also means, off-manifold outliers present in the data are likely to be clustered in with the rest of the data in the embedding space.) This assumption can counteract some of the effects of the curse of dimensionality, specifically the concentration effect.
Like in t-SNE, there are stochastic elements in the dimension reduction algorithm leading to non-deterministic results. Again, there is no guarantee the best result will be found, nor that the resulting structure actually captures a structure present in the data.

4.3.1 Computation

UMAP constructs a union of fuzzy simplicial sets (one for each data point) to represent the assumed manifold, and then converts it to a fuzzy graph, which can be compared to a corresponding low-dimensional fuzzy graph, representing the embedding space. The comparison is made by measuring the distance between the two graphs using cross-entropy, and then optimizing over it.
The two phases using graphs consist of the high-dimensional Graph Construction and the low-dimensional Graph Layout (McInnes et al., 2018, p.14 ff.). The Graph Layout is optimized to approximate the Graph Construction as closely as possible. In the Graph Construction, a weighted k-neighbour graph of the original data is constructed and made symmetrical. The weights on the edges of this graph can be interpreted as the probability that the edge (or one of the directed edges) exists. Then, in the Graph Layout, a low-dimensional representation of the data is constructed. The embedding is initialized with a spectral layout and optimized so that the desired characteristics of the k-neighbour graph are preserved.

Graph Construction For each data point x_i, the k nearest neighbours $x_{i_1}, \ldots, x_{i_k}$ are computed using the (user-defined) dissimilarity measure d.
For each x_i,

$$\rho_i = \min\{d(x_i, x_{i_j}) \mid 1 \leq j \leq k, d((x_i, x_{i_j}) > 0\}$$

describes the minimal distance to another data point.
Now σ_i can be determined s.t.

$$\sum_{j=1}^{k} \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right) = \log_2(k).$$

For any given data point x_i, the associated weights for the k nearest neighbors $x_{i_j}, 1 \leq j \leq k$ can now be set as the terms of the aforementioned sum, therefore

$$w((x_i, x_{i_j})) = \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right).$$

Note that due to the definition of ρ_i, at least one point x_{i_j} exists where the weight equals 1.
Let $\overline{G}$ be the weighted directed graph with the data points as vertices, directed edges between each of the data points, and weights w.
The final graph G shall be an undirected, symmetrical variant of $\overline{G}$. Let G be an undirected weighted graph where the vertices are the data points. The weights on the edge between two given data points x_i, x_j are then

$$w^*(x_i, x_j) = w((x_i, x_j)) + w((x_j, x_i)) - w((x_i, x_j)) \cdot w((x_j, x_i)).$$

Graph Layout In the graph layout phase, a low-dimensional representation of the data is constructed. First, the embedding is initialized using a spectral layout. Then, the UMAP algorithm optimizes the initial representation of the data iteratively as described in the following.
Let $\{\mathbf{y_i}\}_{i=1,..N}$ be the current low-dimensional representation of $\{\mathbf{x_i}\}_{i=1,..N}$.
An edge of the graph G is chosen with vertices x_i and x_j. Since x_i and x_j are k nearest neighbours of each other (at least in one direction), the algorithm moves y_i and y_j closer together using an attractive force. Due to computational restraints, the algorithm chooses

one of the edge's vertices to be moved further away from a sampling of other vertices. For each of those other vertices a repulsive force is used.
Both the attractive and the repulsive force "are derived from gradients optimising the edge-wise cross-entropy between the weighted graph G, and an equivalent weighted graph H constructed from the points $\{\mathbf{y_i}\}_{i=1,..N}$" (McInnes et al., 2018, p.17).
The attractive force between two low dimensional coordinates y_i and y_j is given by

$$\frac{-2ab\|\mathbf{y_i} - \mathbf{y_j}\|_2^{2(b-1)}}{1 + \|\mathbf{y_i} - \mathbf{y_j}\|_2^2} w((x_i, x_j))(\mathbf{y_i} - \mathbf{y_j}),$$

where a and b are hyperparameters. Note that $w((x_i, x_j)) = 0$ if x_i and x_j are not in the same k neighbourhood. This means that there is no attractive force between such points.
The repulsive force between two points y_i and a randomly selected y_j is

$$\frac{2b}{(\epsilon + \|\mathbf{y_i} - \mathbf{y_j}\|_2^2)(1 + a\|\mathbf{y_i} - \mathbf{y_j}\|_2^{2b})}(1 - w((x_i, x_j)))(\mathbf{y_i} - \mathbf{y_j})$$

with $0 < \epsilon \ll 1$ added to the divisor in order to avoid division by zero. Note that the repulsive force is designed to be strongest when x_i and x_j are not in the same neighborhood as the term $(1 - w((x_i, x_j)))$ becomes 1. If x_i and x_j happen to be each others nearest neighbours the weight $w((x_i, x_j))$ is necessarily equal to 1 and the whole term becomes zero. In this case, there is no repulsive force at all between the respective low-dimensional representations y_i and y_j.
"Convergence to a local minima is guaranteed by slowly decreasing the attractive and repulsive forces in a similar fashion to that used in simulated annealing" (McInnes et al., 2018, p.17).

4.3.2 Practical application

There are various hyperparameters that can or have to be set by the user. The most significant ones are the number of nearest neighbors, the minimum distance used in the low-dimensional space, the number of target dimensions, and the number of epochs for the optimization process. Further decisions impacting the results are the chosen dissimilarity measure, as well as the chosen initializing layout (a spectral embedding is the default).

Just as with t-SNE, the choice of initialization is consequential for the resulting embedding (see Kobak and Linderman, 2021).
UMAP measures low-dimensional distances in a similar way t-SNE does, but gives the user control over the minimum distance, or in other words how closely packed data points are allowed to be. This is mainly an "aesthetic parameter, governing the appearance of the embedding" (McInnes et al., 2018, p.23). One can choose a bigger minimum distance to better see local structure, and when clear visual cluster separation is less important.
While the goal with UMAP is to preserve both local and global structure, there is no guarantee that the 'found' structure is actually present in the high-dimensional data, or that it approximates the underlying manifold. UMAP can be run two or even multiple times sequentially to make the structures determined by the method more apparent. But because of aforementioned issues, those results have to be taken with a grain of salt and carefully evaluated and checked.
When using UMAP on less than 500 observations, it might result in suboptimal embeddings due to the number of approximations being made by the algorithm (see McInnes et al., 2018, p.49).
Based on the results in this thesis (see section 7), I would argue that the setting for the number of neighbors in UMAP can have serious effects on the result, especially when choosing a very small neighborhood size.

5 Comparison of embedding methods

No definitive claim can be made as to which technique might work for what kind of data or objective.
While there are some methods to quantitatively evaluate the embeddings, they are not universal and make it hard to compare between embedding methods. If the underlying structure of the embedded data is unknown, it is especially hard to objectively evaluate the embedding quality. Comparisons even between similar algorithms as UMAP and t-SNE are difficult, and it is hard to ascertain which hyperparameters matter or how they correspond between methods (see Wang et al., 2021, p.2).
When embedding to lower dimensions, some information loss is inherently necessary. One of the trade-offs taken is the preservation of global and local structure. UMAP and t-SNE usually work well on local structure, but compromise on or even ignore the global structure (see e.g. Wang et al., 2021). The choice of initialization method is important for both t-SNE and UMAP, specifically for preserving more of the global structure (see e.g.

Kobak and Linderman, 2021). For example, due to the repulsive forces in the algorithms, groups of data points tend to be split up when randomly initialized. Böhm et al. (2022) argue that the attraction-repulsion algorithm via negative sampling is the main source of variability in the resulting embeddings in dimensionality reduction methods using k nearest neighbor graphs, such as t-SNE and UMAP.

Chari and Pachter (2021) point to the lack of theoretical or practical support for t-SNE or UMAP accomplishing the preservation of local or global structure in single-cell genomics in particular, complex data which they deem to be intrinsically high-dimensional. Structures present in 2D or 3D embeddings of such data might therefore well be arbitrary.

Various benchmark comparisons on different (potentially intrinsically high-dimensional) biological datasets exist, mainly with clustering or differentiation objectives. While in most, UMAP comes out on top (see e.g. Yang et al., 2021, Wu et al., 2019, Hozumi et al., 2021) and t-SNE outperforms the other methods in others (see Xiang et al., 2021), the results for the two methods are usually somewhat similar, especially when also compared to MDS. When comparing the techniques on outlier detection tasks for functional data, MDS clearly outperforms UMAP (see Herrmann and Scheipl, 2021, p.988), regardless of the neighborhood size chosen for UMAP.

Taking all of the above into account, it might be inferred that for outlier detection, MDS is generally the better suited technique, while for clustering (especially if there is a need for clear separation), UMAP and t-SNE are usually more helpful. For a smaller number of observations, MDS might be preferable to t-SNE or UMAP, which usually need a bigger sample size to yield consistent results. In practice though, the choice of embedding method has to be made taking into account not only the objective, but also the specific data at hand, possibility of hyperparameter tuning, computational restraints, and more. For more reliable insights, a combination of methods might certainly be the best option. Special care has to be taken with the results of non-linear dimensionality reduction methods such as t-SNE or UMAP, as they might portray structure that is non-existent in the actual data, i.e. they can sometimes be prone to ‘hallucinating’.

6 R Shiny apps for embedding visualization and exploratory analysis of high-dimensional data

6.1 Motivation

As described in the previous chapters, visualization of a dataset is one of frequent motivations for using embeddings. There are countless reasons one might want to visualize their data. Among them: as a first (or intermediary) exploratory step, to get to know the data, to find unknown structures, and build or cement hypotheses about the data. Visualization can also be helpful for clustering or outlier detection. The R Shiny apps developed for this thesis aim to facilitate the visualization process and to be a tool to enable both intuitive understanding as well as targeted analysis. One might for example want to quickly highlight different variables describing the embedded data to visualize relationships or clusters. It is also possible to see directly which data point in the embedding corresponds to which actual data point. Seeing what the actual data looks like can be hugely beneficial for the user. If the embedding space contains more than two or three dimensions, there have to be strategies in place for the user to be able to get an intuitive visual understanding. Furthermore, one might have used multiple different embedding methods or settings for the same dataset and might want to quickly and comprehensively compare the results. The apps aim to make it possible to gain a quick and intuitive understanding of both dataset (functional data or image data) and embeddings via various visualization alternatives and functionalities offered. I have generated most of the plots in section 7 within the apps, in part to demonstrate the functionality.

6.2 Functional data app

6.2.1 Functionality

Dimensionality reduction techniques allow the visualization of complex data like functional data. Given a dataset of functional data and its embeddings, this app offers a variety of visualization options to handily explore high-dimensional functional data. The three different visualization tabs offer a variety of options to display the embedding scatterplots and the corresponding curves. The first tab, ***Viz: Functional data***, plots a single plot of two chosen embedding dimensions, and offers a wider variety of plotting options. The tab ***Viz: Embeddings*** plots all dimensions of all given embedding methods against each other. This gives an overview of the results of all the given embeddings. The

tab ***Viz: Matrix*** plots all dimensions of a chosen embedding method in a scatterplot matrix for intuitive visualization of embedding dimensionalities that are higher than 2D. The tab ***Data*** shows the currently loaded dataset and contains the option to upload another dataset, while the tab ***About*** consists of a general description as well as background and contact information. Each tab offers a description of the functionalities by clicking on 'More information'.
The aim of the app is to enable the user to familiarize themselves both with the embedding methods used and their realizations, as well as with functional data itself, with the assistance of the embeddings. This is achieved by gaining an overview over the embedding dimensions and methods in the *Viz:Embeddings* and *Viz:Matrix* tabs. For a more detailed inspection of a chosen scatterplot, the *Viz: Functional data* tab is better suited. All of the visualization tabs allow the user to make selections within the scatterplots and see the curves belonging to the scatterplot points, as well as read out the IDs of said curves. Moreover, the user can compare the location of the selected point in different embedding dimensions and methods, to answer questions like: 'Are these functions generally grouped together?' or 'Is this function distant from the bulk of observations in most embeddings?'. Additional information contained in the dataset can also be visualized with the help of colors, shapes or labels. This enables the user to e.g. select scatterplot points based on a variable value. There is also an option, to directly plot small representations of the curves in the embedding scatterplot instead of points.
That way, structures both in the embeddings as well as the curves can be discovered, made apparent, or simply conveniently plotted.
For a detailed description of all features, possibilities and requirements, see subsubsection 6.2.2 and subsubsection 6.2.3.

6.2.2 Input data

The data to be uploaded has to follow a specific format. This makes a certain amount of preprocessing inevitable. The app requires an R data frame, saved as a R-object RDS-file. The data frame has to have a specific structure and naming convention, to be usable in the app. To save and display the functional data, the R package `tidyfun` (Scheipl et al., 2022) is used. I have provided a detailed description of the required format in Appendix A.
The preloaded example data consists of a selection of mouse hearing curves described in subsubsection 7.1.1.

6.2.3 (Interactive) visualization

In the following, I will give detailed descriptions and instructions for each of the visualization tabs.

Viz: Functional Data

This tab makes it possible to plot two chosen embedding dimensions in a scatterplot and to interact with it in a variety of ways. For first time users, short instructions are given at the top of the page, including a link to the example data source. These instructions change depending on the action taken. To explore the visualization possibilities, a click on the 'Plot the plots' button before uploading new data uses the example dataset. New data can be uploaded in the *Data* tab.

When plotting the embedding scatterplot in the top plot, the represented functions are plotted in the bottom plot. If the check box for 'Plotly' has been checked, there is a third plot, showing another version of the embedding scatterplot, with additional in-plot options, e.g. zooming in and out or displaying points by variable.

By hovering over a point in the 'Embedding Plot' , the corresponding curve is highlighted in the 'Functions' plot. Additionally, a tooltip with information on the hovered function is displayed, including the ID and the embedding dimension coordinates. When moving the cursor out of the plot quickly after hovering, the previously hovered point will stay selected. To de-select, one simply has to move the cursor to an empty space within the plot. The functionality can be disabled by clicking the 'disable hover' check box. This can be useful when hovering on points accidentally slows down the app due to prolonged plotting times, e.g. when there are a lot of data.

For highlighting multiple curves in the 'Functions' plot there are the foloowing options:

- Choosing a collection of points by brushing (via click and drag) in the top plot. All curves represented by the brushed points are highlighted. To undo the selection, one can brush in an empty space in the plot.
- Choosing all points of a certain variable value indicated by color by double clicking on a point. All curves with that variable characteristic are highlighted, as well as the selected scatterplot points.

The IDs of all selected points, be it via hovering, brushing or double clicking, are printed in the box beneath the 'Functions' plot.

By clicking 'Plotting Options' one can change the visual depiction of the scatterplot:

- Choosing the embedding dimensions to be plotted.
- Coloring the points according to a chosen variable (continuous or categorical).
- Labeling the points with the value of a chosen variable.
- Differentiating the points by shape according to a chosen variable (categorical with six or less categories).
- Plotting small curve plots instead of points in the scatterplot by checking 'Glyph plot'.
- By checking 'Plotly', including a third plot (see below).
- Entering a title in 'Custom scatter plot title', to change the title shown in the scatterplot.
- Entering a title in 'Custom functions plot title', to change the title shown in the plot showing the curves.
- Using the slider to set the size of the points in the scatterplot. This can be be helpful in case there are too many points to differentiate, for example.
- Changing the 'Color to highlight double clicked points'. These points might be more visible, depending on the variables chosen to be represented by color.

All variables chosen in 'Plotting Options' are indicated when hovering on a point in the scatterplot. This is true even when 'Plot the plots' has not been clicked after changing variables. This enables the user to include variables in the tool tip without overbearing the scatterplot with information.
The extra plot, available when 'Plotly' is selected, makes it possible to zoom in and out of the plot. It also allows variable selection: hide or display points belonging to a variable by clicking on said variable in the legend. Double clicking on it will show only points belonging to that variable. This can be very useful for revealing if points of one variable are grouped together or stretched out in dense areas of the plot. It also lets the user select 'Compare data on hover' which displays information on all points close to the hovered point. This is especially helpful, when there are multiple points overlaid. (The zoom feature also helps in that case). There is also a 'Download as a png' button.
The structure of the matrix visualization was inspired by Herrmann and Scheipl (2021).

Viz: Embeddings

This tab plots all given embedding dimensions of corresponding methods against each other. It is necessary to click on the 'Plot the plots' button to plot them, using the example dataset if no new data has been uploaded. That way, the visualization options can be set, before plotting a potentially rather big number of plots.
Each possible combination results in a trio: On the left, the embedding scatterplot. On the right, the represented functions are plotted when hovered on or brushed. Beneath, there is a field with information on the IDs of brushed and hovered points.
By hovering over a point in the left scatterplot, the corresponding function is highlighted in the function plot. For highlighting multiple functions in the function plot, one can choose a collection of points by brushing (via click and drag) in the left plot. All curves represented by the brushed points are then highlighted.
There are again several options for changing the visual aspects of the scatterplot:

- Coloring the points according to a chosen variable (continuous or categorical).
- Labeling the points with the value of a chosen variable.
- Differentiating the points by shape according to a chosen variable (categorical with six or less categories).
- Checking 'Glyph plot' will plot small curve plots instead of points in the scatterplot.
- Using the slider to set the size of the points in the scatterplots to make them more easily differentiable or visible.

To see a legend with information on e.g. the colors used in the scatterplots, one has to brush in one of the scatterplots, as the legend is plotted within the curve plot.
For my analyses I used this tab to get a quick overview over all included scatterplots, to identify peculiarities, similarities, differences or even mistakes, before taking a closer look at specific embeddings. It is also useful for making differing selections in different scatterplots, and being able to compare the highlighted curves, as they are each plotted in their own plot beneath one another.

Viz: Matrix

This tab plots all given dimensions of a chosen 'Embedding method' against each other. The plots are arranged as a scatterplot matrix, to facilitate insights into spatial patterns.

All the curves represented by the points in the scatterplots are plotted above in a separate plot. The possibilities are:

- Choosing a collection of points by brushing in the first plot. All curves represented by the selected points are highlighted. Beneath the plotting options on the left side, there is a field with information on the IDs of brushed points.
- Changing the plot to select points in. The drop-down menu called 'Plot to brush' lets one select the desired plot number.
- Seeing the position of a selection of points in other embedding methods. When changing the 'Embedding method', the chosen points will still be highlighted.
- Undoing the brushed selection and seeing the original plots by clicking the button 'Undo brush'.

There are a couple of options for changing the visual aspects of the scatterplot. They can be accessed by clicking 'Plotting Options':

- Coloring the points according to a chosen variable (continuous or categorical).
- Differentiating the points by shape according to a chosen variable (categorical with six or less categories).
- By changing the 'Color to highlight brushed points', these points might be more visible, depending on the variables chosen to be represented by color.
- Using the slider to set the size of the points in the scatterplots to make them more easily differentiable or visible.

It is possible to choose a selection of points, and then change the embedding method, to see where they have been embedded in the other methods. This comes in handy for comparing between methods, and checking if patterns are reproduced independent of the method.

General comments
If the curves take too long to load, it might be advisable to upload a sparser grid, i.e. to for example only include every third measurement.

6.3 Image data app

As this app follows the design of the functional data app very closely, I only mention the parts where the two apps diverge in this section. For anything not described here, refer to the functional data app section.

The data to be uploaded follows the same pattern as the functional data, instead of the functional measurements an url can be included, to show the pictures. To visualize any embeddings, without including the images, this information can be excluded.
In the *Viz: Image Data* tab, instead of the lower plot three randomly chosen example images from the dataset are shown beneath the scatterplot.
When choosing 'Plot images', a miniature version of the image is plotted in the scatterplot instead of points. The background color of the plot changes as well, to make sure a white background within an image is visible.
The brush option in the scatterplot via click and drag only returns the IDs of the brushed images, as it would be difficult to show an arbitrary number of chosen images simultaneously. When hovering on a point, additionally to the tool tip, the hovered image is plotted next to the example images.
There is no 'Disable hover' button. As only one image has to be plotted on hover, it does not take as much time when accidentally hovering on a point. In my opinion, this option is therefore not necessary.
'Example images by:' gives the user some control over the selection of the example images, being able to choose a variable, to ensure that the images shown are not all chosen from the same category (only categorical variables can be chosen as an option here).

In the *Viz: Embeddings* tab, the images representing the points hovered on, are plotted to the right of each scatterplot. Here too, it is possible to plot the images instead of points in the scatterplots.

In the *Viz: Matrix* Tab, it is possible to hover on the scatterplots (choosing the appropriate plot first). As the images will be shown beneath the plots, they might not be immediately visible. Scrolling down will show the image, a new one is only rendered when hovering on a different point.

6.4 Future optional adjustments

There are several possibilities to expand on the current versions of the apps, to make them more broadly applicable, or better suited for very specific visualization needs or to simply improve them. As the current versions are geared towards functional and image data, they could be expanded to support other kinds of high-dimensional data, like graphs, biomedical data or tabular data in general. Other features not yet included are incorporating an interactive 3D plot and an option to save and compare different plots within the app. Furthermore, the dimensionality reduction methods have to be applied before the usage of the app. To be able to embed the dataset in-app could be an option as well. These are simply a few examples of possible further alternative implementations. Furthermore, the apps have not been extensively tested by users other than myself. It is to be expected, that further adjustments for practicability will arise when being used for real-world application.

7 Exemplary analysis on real life data

7.1 Functional data: ABR curves

7.1.1 Description of dataset

As an example for functional data, I used auditory brainstem response (ABR) measurements performed on over 4000 mice, collected and provided by Thalmeier et al. (2021). Some of the mice are wildtypes, the majority are mutants of a single-gene knockout line. To test the hearing, the brainstem voltage in reaction to a sound has been measured for 10 milliseconds. Each measurement is labeled among others with the mouse ID, the frequency the sound was played at, the sound pressure level of the sound, the knockout gene, and the result of their automated detection method, i.e. if the mouse heard the sound. For this thesis I only used measurements where the mouse did hear something. The frequencies used are $6000hz$, $12000hz$, $18000hz$, $24000hz$ and $30000hz$. In addition, there was a broadband noise used, encoded in the dataset as frequency 100. The sound pressure levels (SPL) within the dataset after eliminating measurements without any discernible reaction, range from $10db$ to $90db$, with increments of five decibel. The data has originally been collected to study the relationship between the mouse geno- and phenotype with regards to hearing ability. As the collection process was meticulous, target-oriented and extensive with a lot of observations, the dataset is well suited for various kinds of

analyses. Although automated processes for the labeling of ABR curves are an active field of research, in standard ABR analyses, characteristic peaks of the curves are manually labeled by trained professionals (see e.g. Krumbholz et al., 2020, p.2).
To use the datafor comparing the embedding methods and visualizing the measurements as curves, I made minimal adjustments to the available raw data. The voltage was measured at almost 2000 time steps within the 10 milliseconds. Instead of using all of the time steps, which necessarily include a lot of un-informative noise, I used only the first 666 steps. To be able to compare meaningfully, I used smaller selections of curves based on certain characteristics, e.g. only curves resulting from a tone of frequency 12000.

Preloaded example data

The data preloaded in the app consists of the hearing curves of two mice ranging from frequency 100 (broadband) to 30000, with sound pressure levels ranging from $20db$ to $85db$. It contains a 5D MDS embedding and a 2D UMAP embedding of said curves, as well as LOF-score variables. The MDS LOF-scores given in that dataset are calculated on the 5D embedding, the UMAP LOF-scores are calculated on the original data, as the distances in a 2D embedding are more easily recognizable. UMAP is also less likely to preserve global distances present in the data, making the LOF-scores on the original data more informative.

7.1.2 Results

To not overload the thesis, I describe only a few results of the exemplary data analysis in this chapter. All the described datasets are included in Appendix A and can be explored in the functional data app.

MDS

As described in subsection 4.1, a standard practice to decide on the MDS embedding dimensionality is to look for an elbow when plotting the GOF-scores as seen in Figure 3. If there is any elbow to be assumed, it would occur after the third dimension. Various practical applications use a measure of a GOF > 0.8 as a threshold value. There is no inherent reason to choose that value, but as five dimensions are still well suited to be visualized and other curve selections showed even less of a drop in increase I decided on five embedding dimensions as a fitting number for the MDS embeddings of these data.
The preloaded data consists of embeddings of a small dataset containing only a few curves with large variability. The first two MDS dimensions exhibit clear structure, which

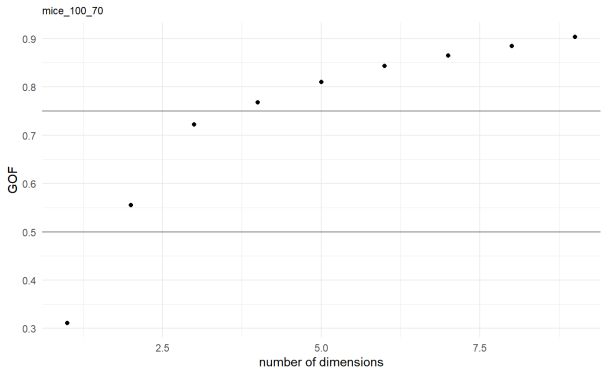

Figure 3: The GOF values for classical MDS embeddings of mouse hearing curves to clicks of 70db up to 9 dimensions

corresponds to the information we have on the curves, as can be seen in Figure 4. The two long 'tentacles' to the left of the bulk of the data belong to the broadband curves (frequency 100), each of them to one of the mice. The further away from the center of the bulk of data a point lies, the higher the sound pressure level, as indicated by the labels. Within the bulk of the data, we can observe a similar pattern, with lower sound pressure levels, and higher frequencies tending to be huddled towards the center. This aligns with our knowledge of the data. If the noise was louder or the frequency lower, the tone is usually easier to hear, meaning the peak amplitude will be larger and therefore also visually more detectable. The curves are usually especially pronounced with frequency 100 (broadband click) and higher sound pressure levels, as can be seen in the 'Functions' plot.

When embedding the larger datasets with MDS, it was harder to recognize structure other than more central vs. more outlying data points within the MDS embeddings. Coloring the data points by variables was at times essential to identify patterns. Different embedding methods (classical vs. non-metric) and different distance metrics (L2, L1, Canberra) yielded only marginally varying embedding results.

t-SNE

The t-SNE embedding into two dimensions on the ABR mouse-curves offers potentially valuable insights even without hyperparameter tweaking, just using the default t-SNE settings. As can be seen in Figure 5, it separates the noise curves (black) from the pure tone curves almost perfectly. The curves with frequency 6000 (purple) form a visible cluster as well. The rest of the frequencies do not separate in the first two embedding dimensions, but there is clear structure to be seen. There is a small extra cluster forming

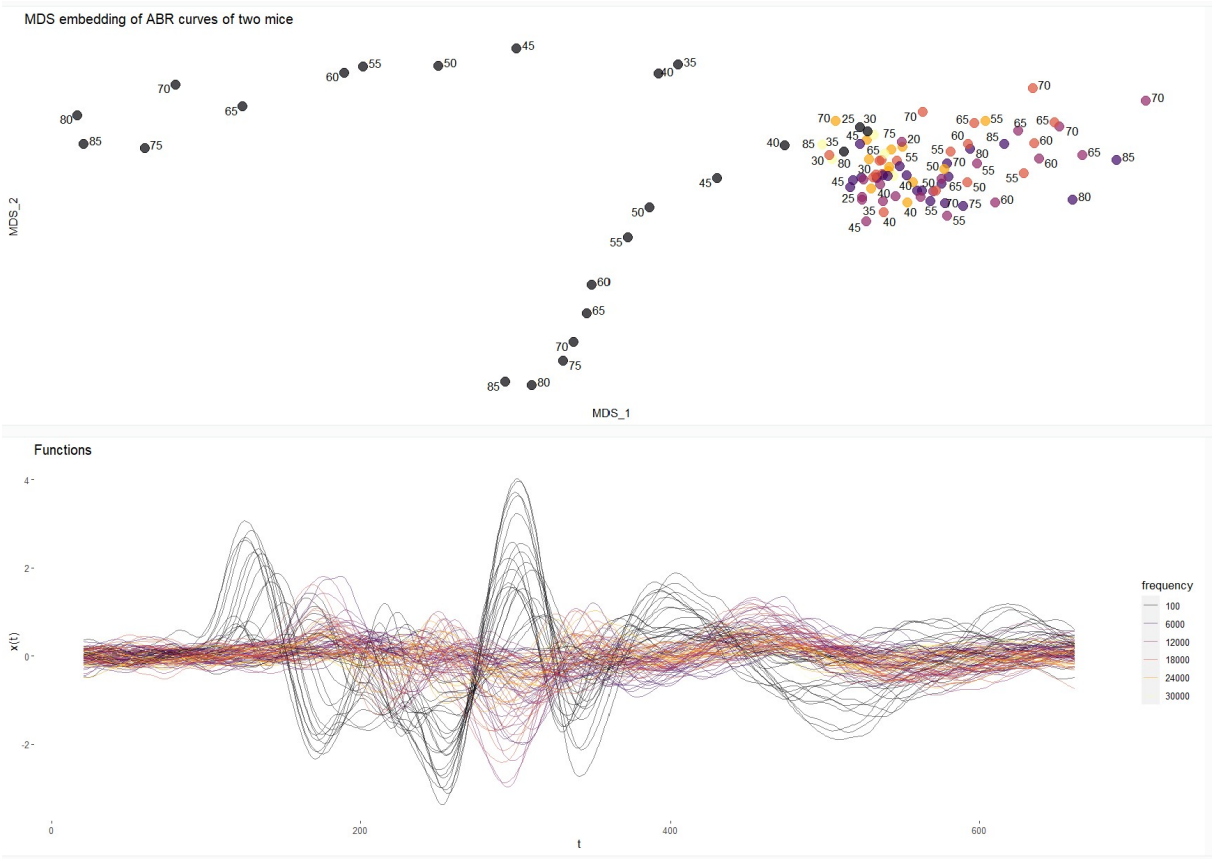

Figure 4: MDS embedding and ABR curves of two individual mice

on the right side which is not associated with the frequency. When looking at the gene labels for this little cluster, there seems to be a connection, as can be seen in Figure 6. The noise data points clustered in with the tone data points in the upper middle also carry this gene label. Within the frequency 6000, the data points with this knockout gene still form part of their cluster, but they can be found clearly on the far left side of it.
When taking a look at the curve plots for each of these groupings, the embeddings of each of these seem to instinctively make sense. As can be seen in Figure 7, the curves generated by noise rather than a pure tone of a certain frequency (represented by black and the number 100), are, when given the relevant information, easily recognized to follow a different pattern to the rest.
In Figure 8 all the curves belonging to mice with the knockout gene Fam53b are highlighted. Given the information just obtained from Figure 7, it is reasonable to assume that those curves standing out from the bulk due to their peaks occurring earlier and generally having a higher amplitude probably have been generated by noise.
This suspicion is confirmed when comparing the curves in the mini cluster as seen in Figure 9 with the noise cluster in Figure 10. Less obvious and barely visible, much less separable by sight, are the curves generated by frequency 6000. When only highlighting those curves, as done in Figure 11, their differing characteristics become apparent. Not only do their peaks have a delayed latency and lower amplitude compared to the other curves, they are also quite similar to each other. Paying close attention, it is perceptible that not all Fam53b-curves are captured by these three groupings. In Figure 12 we can see that there are a few curves of that variable grouped in with the bulk of the observations. When coloring embeddings by variable, it can be easy to overlook such hidden data points, especially if the plots are colored by variable, with overlying points concealing points lying beneath. This can make clusters and grouping appear more uniform than they actually are.
There are some data points, whose position in the embedding do not correspond to the information available. There are a few curves from other frequencies embedded within the upper right of the space of the noise curves for example. There is also some structure visible within that cluster not associated with the given variables. Nevertheless, when looking at the curves in question, the structure does seem to exist in the data.
While not every aspect of the structure in the embedding seems intuitively connected to the variables of interest, it can be said that the embedding returned by t-SNE in this case does aptly capture a lot of the structure of the ABR mouse curves.
Changing the hyperparameter settings (i.e. no PCA initialization, changes in the perplex-

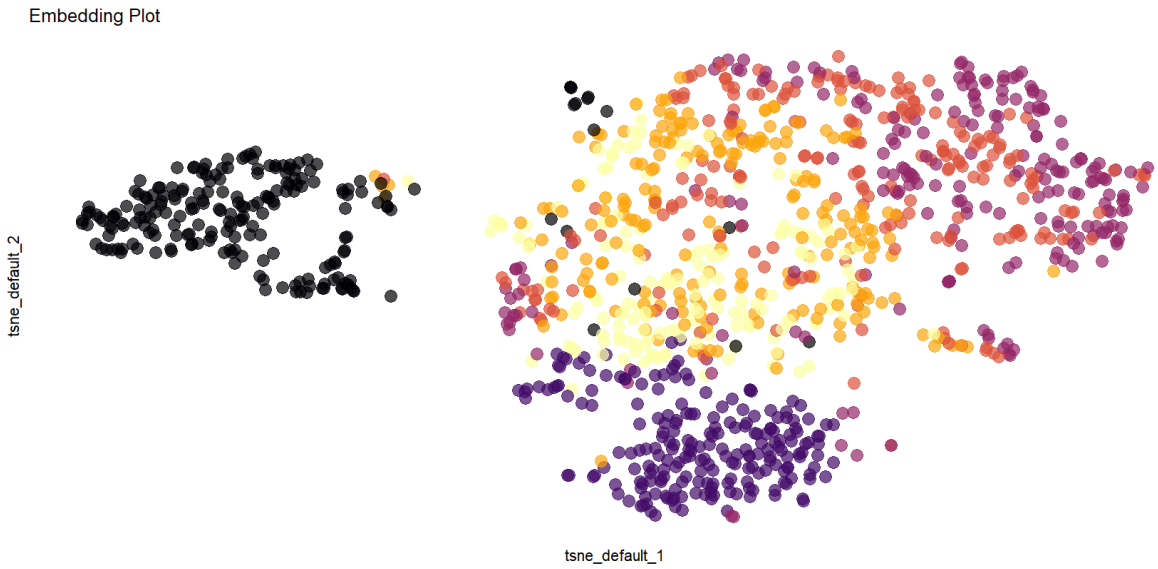

Figure 5: t-SNE embedding with default settings of ABR curves, all with SPL 70, colored by frequency

ity, using pure t-SNE without Barnes-Hut approximation) yielded only minimal changes in the results.

UMAP

The default UMAP embeddings generally performed similarly to the t-SNE embeddings. Other than with t-SNE though, some changes of the hyperparameter settings did have a noticeable impact. For an easy comparison, I chose some examples from the same dataset for illustration. In Figure 13 a 2D UMAP embedding using the default settings is shown. As in the corresponding t-SNE embedding, the most obvious is one clearly separated cluster, mainly consisting of noise curves (frequency 100). While the global appearance differs, a lot of the structure appearing in the t-SNE cluster can be found in this embedding as well. There is a group of mostly frequency 6000 curves almost separating, and a small extra cluster above the points in the middle. As revealed in Figure 14, this is the same group of 24 FAM53b curves already observed in the default t-SNE embedding. Other structures that are reappearing are the few curves of other frequencies grouped in with the black noise curves, as well as the noise curves carrying the Fam53 characterization being grouped in with the main cluster (although not all of those points are Fam53b curves, as can be seen in Figure 14).

Using a random initialization on the dataset, (see Figure 15), shows the group of black noise points being 'split in two'. This is in all likelihood due to the random initialization and attraction-repulsion algorithm employed by the UMAP algorithm. It illustrates the

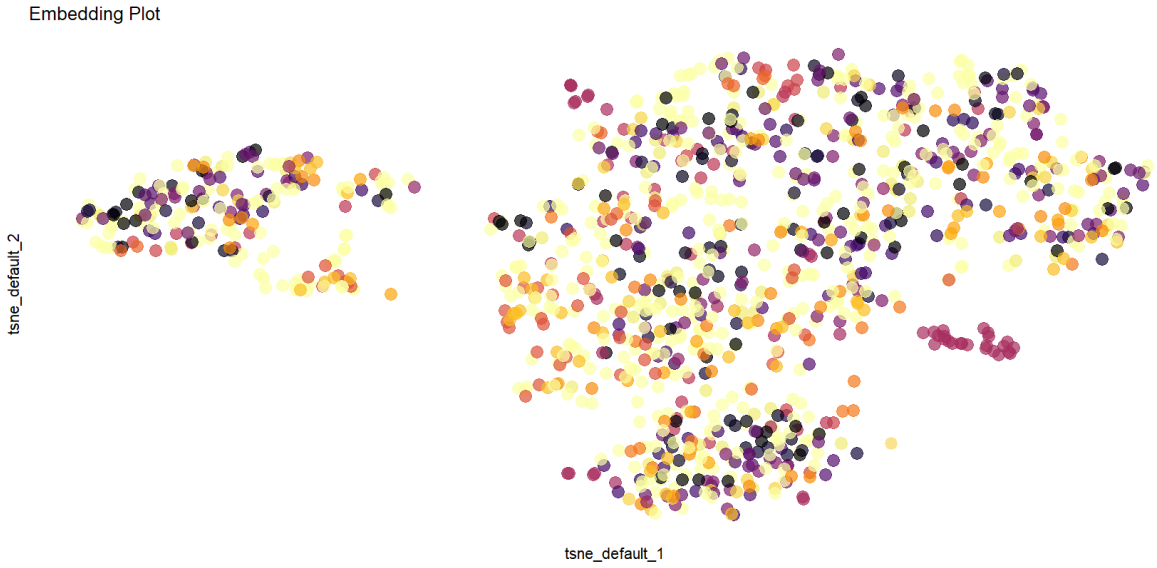

Figure 6: t-SNE embedding with default settings of ABR curves, all with SPL 70, colored by knockout gene

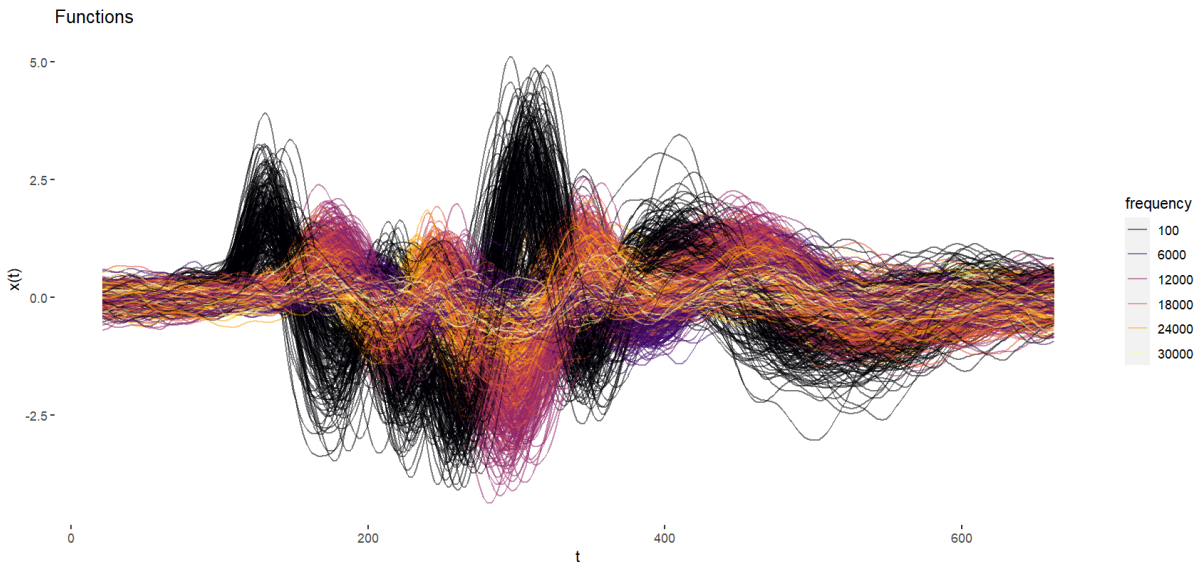

Figure 7: ABR curves, all with SPL 70, colored by frequency

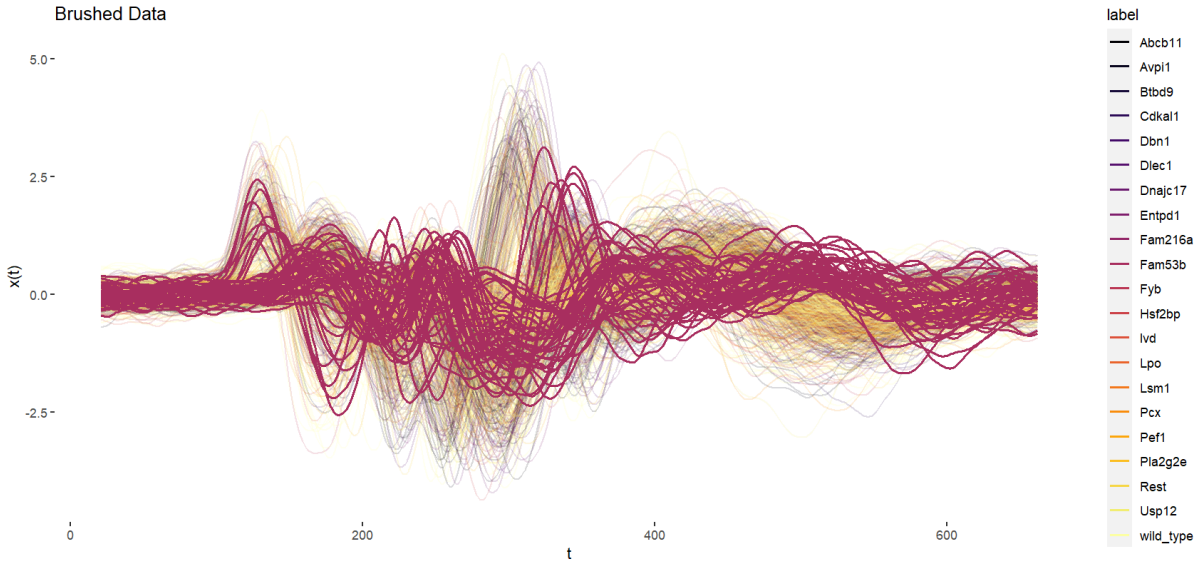

Figure 8: ABR curves, all with SPL 70, colored by knockout gene with Fam53b highlighted

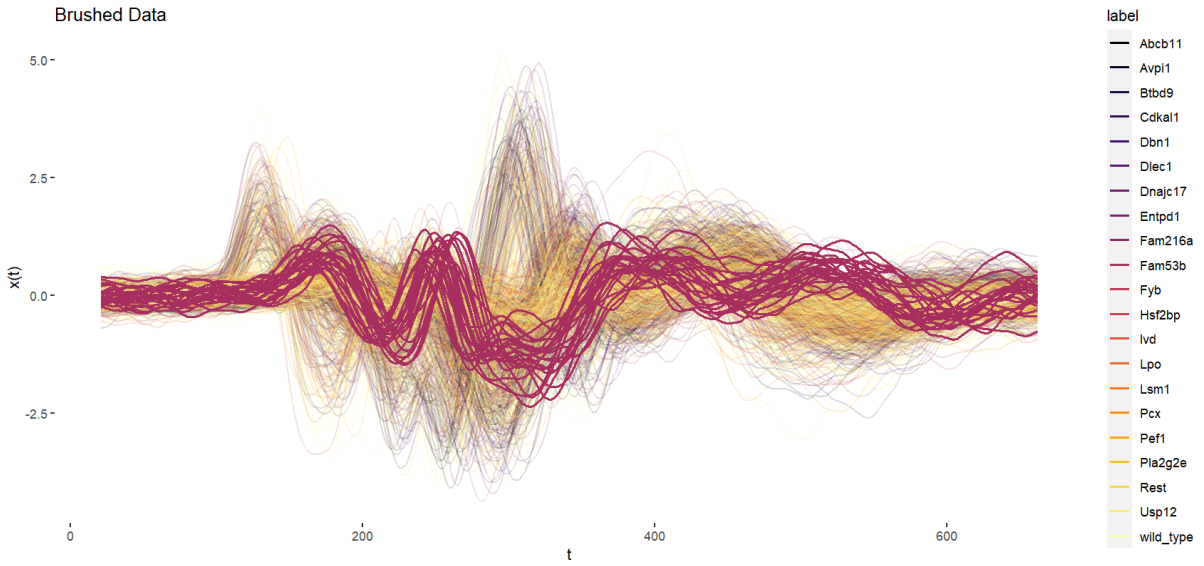

Figure 9: ABR curves, all with SPL 70, colored by knockout gene with Fam53b, frequency 12000 and higher highlighted

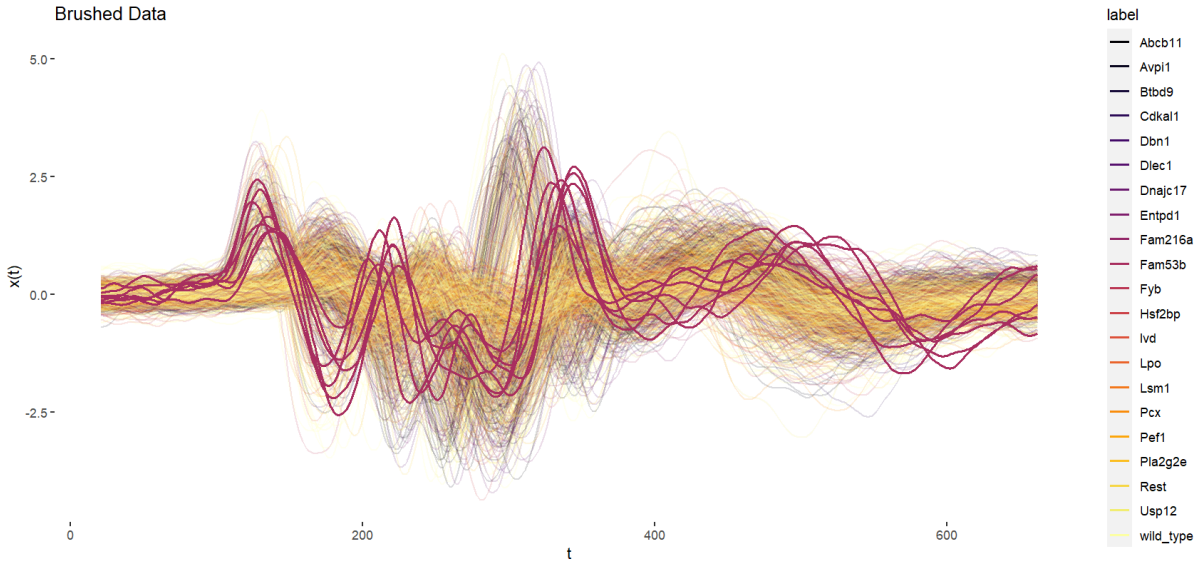

Figure 10: ABR curves, all with SPL 70, colored by knockout gene with Fam53b noise curves highlighted

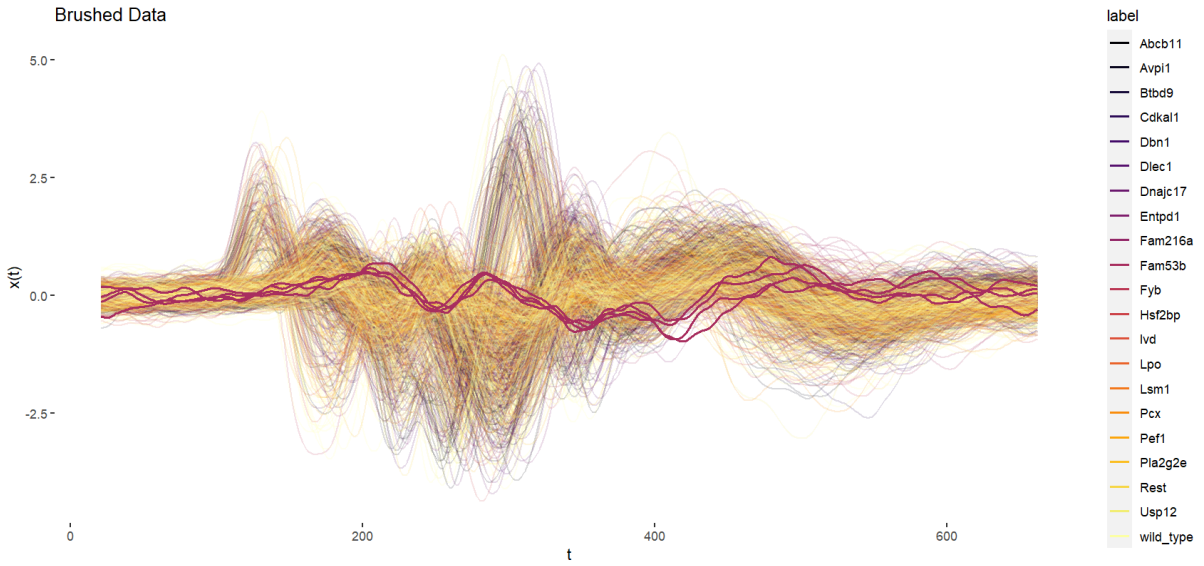

Figure 11: ABR curves, all with SPL 70, colored by knockout gene with Fam53b frequency 6000 highlighted

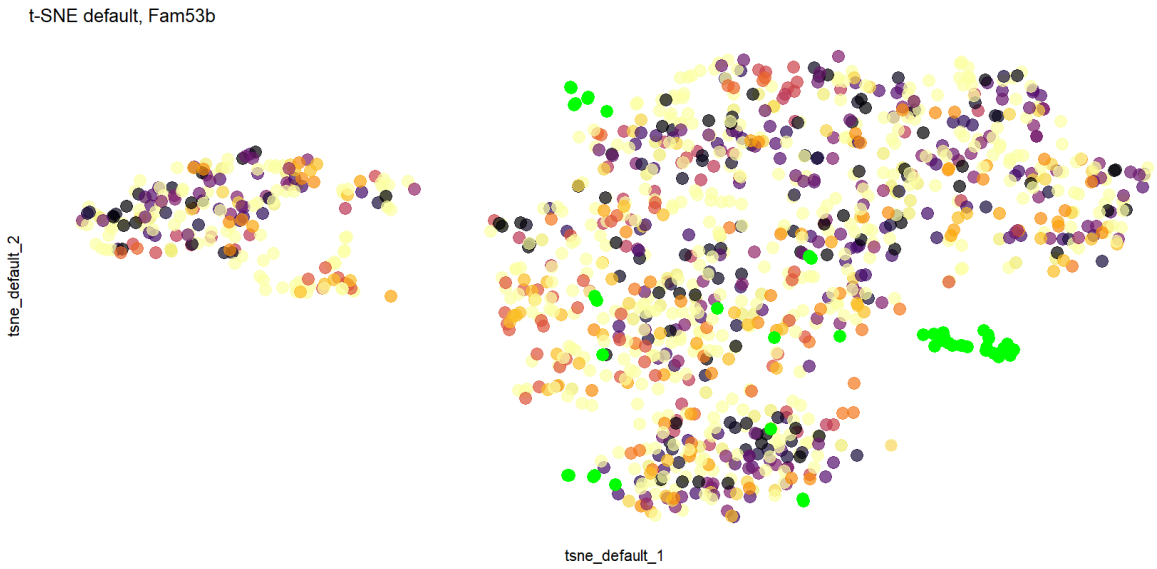

Figure 12: t-SNE embedding with default settings of ABR curves, all with SPL 70, colored by knockout gene, Fam53b in green

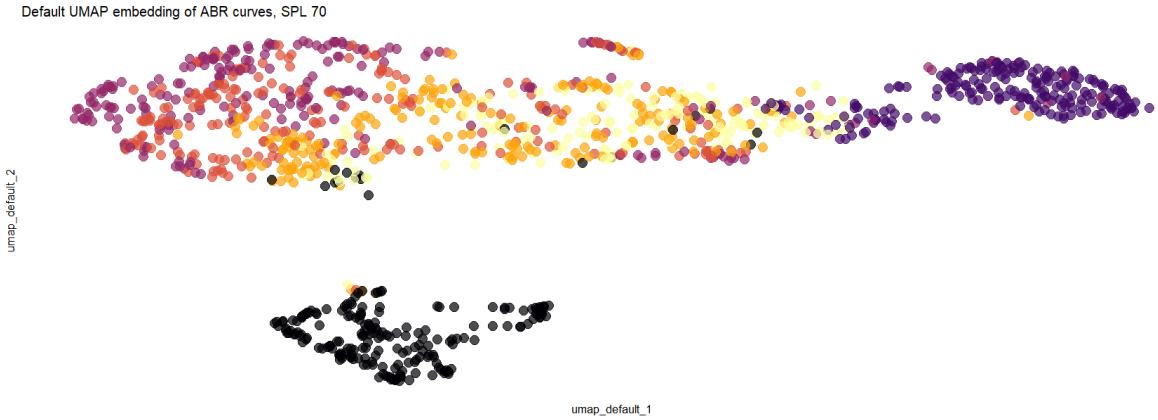

Figure 13: UMAP embedding with default settings of ABR curves with 70*db*, colored by frequency

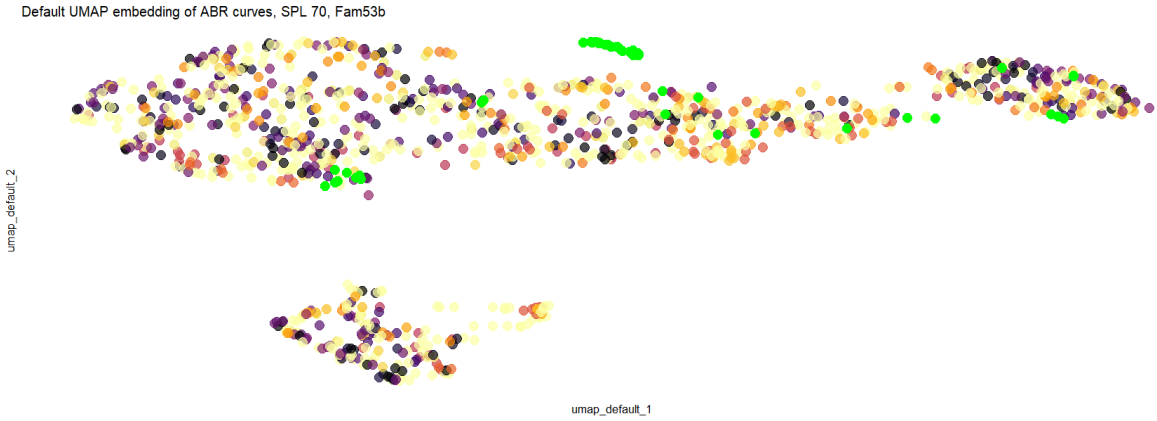

Figure 14: UMAP embedding with default settings of ABR curves with 70*db*, colored by knockout gene, with Fam53b highlighted

Figure 15: UMAP embedding with random initialization of ABR curves with 70*db*

uncertainty inherent in structures shown by UMAP embeddings. It also speaks to the expectation that absolute distances within UMAP embeddings are not necessarily meaningful. The small group in the middle of the plot, selected with the brush as indicated by the grey square, are the now already familiar 24 curves belonging to mice with the knockout gene Fam53b. The black noise points, grouped in the lower part of the bulk of data, carry the Fam53b label as well.

Setting the number of nearest neighbors to a very low number ($k = 3$), skewed the results noticeably. While the effects are somewhat 'squiggly lines' for less observations, as the number of observations becomes bigger, an interesting phenomenon can be observed, as in Figure 16. As can be seen by the brushed points, all lying basically on top of each other, a lot of these mini-groupings form, likely not approximating any structure present in the data, but rather visualizing the UMAP optimization process.

Setting the nearest neighbors to $k = 50$ yield more desirable outcomes. The results seem mostly similar to the default settings of $k = 15$, but with less clear separation of clusters, and a tendency for the points to appear more stretched out.

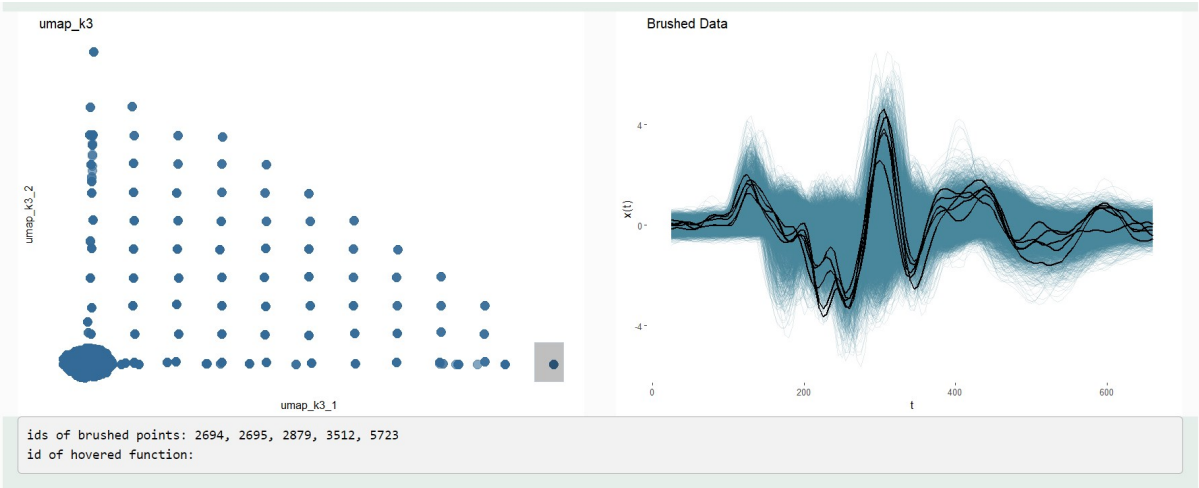

Figure 16: UMAP embedding of 11311 curves, nearest neighbors $k = 3$

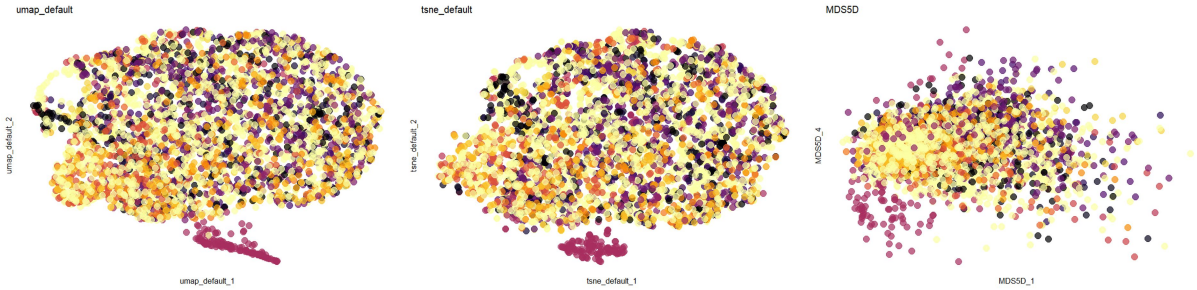

Figure 17: UMAP, t-SNE and MDS embeddings of mouse curves with frequencies $12000hz$ to $30000hz$ and sound pressure levels ranging from $50db$ to $80db$, colored by knockout gene

General remarks

When comparing the embeddings of a relatively big dataset, containing curves of both varying frequencies and sound pressure levels (though no broadband noise or $6000hz$ curves, nor very low sound pressure levels), both t-SNE and UMAP clearly separate a cluster containing Fam53b-curves in most dimensions and with most settings. In MDS there is no clearly separated cluster, but a clear grouping does appear in the fourth dimension, as seen in Figure 17.

In general, the embeddings seem to portray a lot of the structure present in the data, as it appears to be associated with the signal present in the hearing curves. Curves with a visibly strong signal (higher amplitudes, earlier peaks) tend to be embedded close together, with a certain gradient in the embedding space as the signal in the curves gets weaker. Curves identified by the Fam53b knockout gene are grouped or clustered together in most of the embeddings and datasets, although in the MDS embeddings they usually

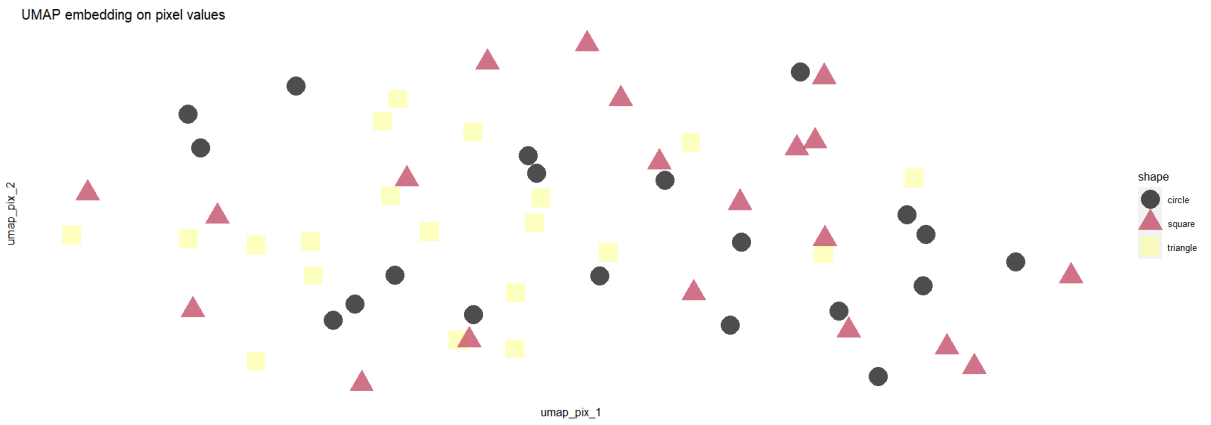

Figure 18: UMAP embedding of the pixel value vectors of hand drawn shapes

only appear in the fourth or fifth dimensions. There are certain shapes the different embedding methods tend towards. t-SNE and UMAP embeddings often form roundish and tube-like shapes, which stretch out and form clear borders. MDS embeddings more often form a dense center, with points feathering out. MDS appears to be more robust towards changes in the number of observations.

7.2 Image data

The example dataset used for the image data app consists of 60 drawings of basic shapes (circles, squares and triangles)[1]. The direct vector representation of the pixel values is not very informative and represents distances between images very unreliably. See subsection 2.3 for a more detailed explanation. Therefore, my goal was to obtain a more informative high(er)-dimensional vector representation for the images, before embedding the image dataset into lower-dimensional space. Unsupervised deep representation learning methods can be used for this end.

An example of this approach can be found in Chuang et al. (2020), where the images are first embedded via debiased contrastive learning (using the SimCLR framework), and then plotted after an additional t-SNE embedding to visualize the learned structure. Applying the same strategy, I used the vision transformer model ViT (see Dosovitskiy et al., 2020) to obtain representation vectors for the images.

The difference this makes in the actual embedding can be seen in Figure 18 and Figure 19. Even on a small number of observations, a much bigger part of the structure is captured by the UMAP embedding on the representation vectors.

[1]The dataset can be found in Appendix A.

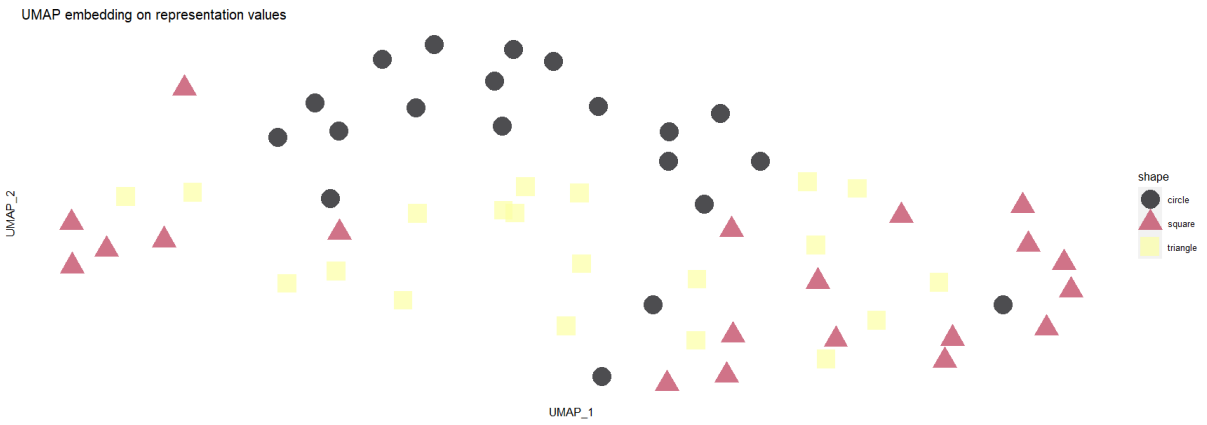

Figure 19: UMAP embedding of the vectors obtained by representation learning of hand drawn shapes

Another image dataset used for comparison containing color pictures of Pokemon was created by Subbiah (2018). This dataset contains an interesting oddity, where some of the pictures have a transparent background, while the others have a white background. This is not immediately visible when inspecting the images visually. Both in the MDS and the UMAP embeddings of the representation vectors, these two types of images were very clearly separated in the first two dimensions, as can be seen in Figure 20. Additionally, there is some local structure in the sub-clusters in both embeddings, which seems to correspond to the size of the creature.

8 Discussion and conclusion

After delving into the challenges and possibilities connected to complex high-dimensional data, this thesis took a closer look at three nonlinear dimensionality reduction methods, namely MDS, t-SNE and UMAP. As these methods are regularly used for and evaluated via visualization, this motivated the development of a set of apps, allowing for interactive visualization of the results of these embedding methods. The resulting R Shiny apps were then used to conduct exemplary exploratory analysis on real-life functional data, as well as some image data.

During the work on the exploratory analysis on the functional and image dataset in section 7, the apps presented within this thesis proved to be a convenient and useful tool. They significantly streamlined the process, enabling not only quick visualization and comparison of the embeddings but also interaction with the results. This facilitated the discovery of structures within the embeddings that were not immediately apparent. Particularly for the functional data, it allowed for the visualization of data grouped to-

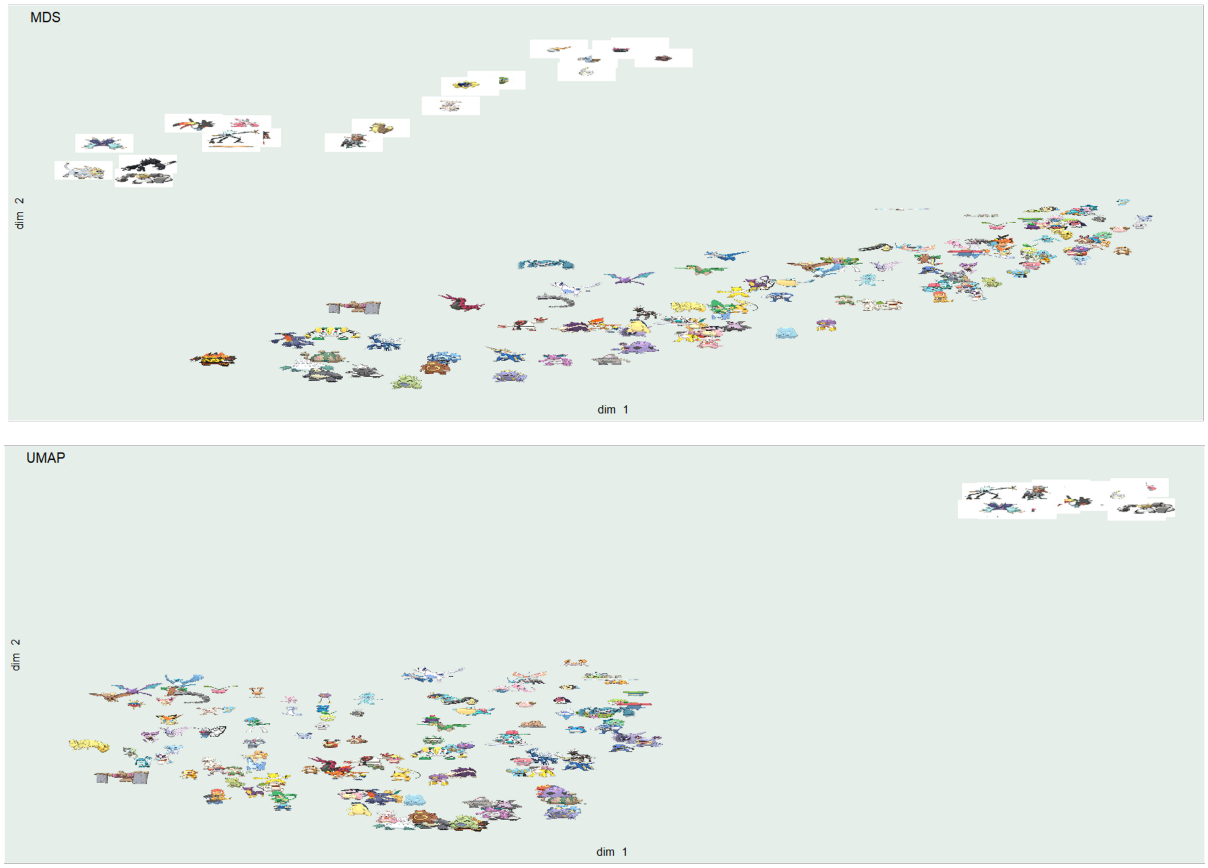

Figure 20: MDS and UMAP embeddings on representation vectors of Pokemon image data

gether in the embeddings, and to form hypotheses about both the data and the embedding process.

While the presented data analysis is not exhaustive, the obtained results, especially on the functional mouse ABR hearing curve dataset, appear promising. Certain structures consistently emerge across many low-dimensional embeddings, irrespective of the method and setting. These structures partly align with pre-existing knowledge about the data, such as curves obtained by a loud broadband click differing distinctly from curves obtained by pure frequencies. Moreover, previously inconspicuous structures were observed in the embeddings. In many of the embeddings, small clusters were revealed, containing only curves by mice with the knock-out gene Fam53b, suggesting an association with the mouse genotype. A more detailed and complete analysis of the results of the different embedding methods, hyperparameter settings and design choices specifically on the functional mouse ABR hearing curve dataset could confirm the validity of this first impression. A deeper analysis could provide more insights into the adequacy of the nonlinear dimensionality reduction methods MDS, t-SNE, and UMAP for this specific type of functional data. The results in this thesis indicate that these methods are suitable for exploratory analysis of such functional data. The experience within the work for this thesis was that the methods employed were more useful for the functional data than for the image data. This was especially true before preprocessing the images via deep learning.

There are known issues when using nonlinear dimensionality reduction techniques. The apps can help mitigate some of the problems associated these methods for exploratory analysis or, at the very least, provide users with the ability to examine the embeddings in relation to these issues. For instance, data separation may sometimes appear more perfect than it actually is. Due to the order of points in the scatterplot being colored, colors belonging to different variables can be hidden beneath the a seemingly uniform selection of points of a different color. By visualizing all points of one color in the plots, these hidden points can be exposed. By visualizing various embeddings and having the ability to directly compare them, users may be more likely to identify spurious clusters if they are only present in individual embeddings.

The choice of the embedding method proves to be crucial, as some methods are more suited for certain tasks than others. For example, t-SNE and UMAP do not lend themselves to outlier detection or the representation of global distances. MDS does often not separate clusters effectively and might require more than two or three dimensions to depict enough of the structure. Nonlinear dimensionality reduction methods are not necessarily reliable representations of the dataset and should be carefully evaluated.

In conclusion, the results obtained through the use of the apps for exploring embeddings of functional and image data suggest the inherent potential in dimensionality reduction methods for handling complex data.

9 Abbreviations

Table 1: Abbreviations

ABR	Auditory brainstem response
CoD	Curse of dimensionality
FD	Functional data
Freq	Frequency
GOF	Goodness of Fit
iid	Identically independently distributed
k	Number of embedding dimensions
LOF	Local Outlier Factor
n	Number of observations
MDS	Multidimensional scaling
p	Number of observed dimensions
PCA	Principal Component Analysis
SPL	Sound pressure level
t-SNE	t-distributed stochastic neighbor embedding
UMAP	Uniform manifold approximation and Projection

A Electronic appendix

Data, code and figures are provided in electronic form at `https://github.com/jukaje/embed-it`.

References

Bellman, R. (1957). *Dynamic programming*, Rand Corporation research study, Princeton University Press, Princeton.

Beyer, K., Goldstein, J., Ramakrishnan, R. and Shaft, U. (1999). When is "nearest neighbor" meaningful?, *in* G. Goos, J. Hartmanis, J. van Leeuwen, C. Beeri and P. Buneman (eds), *Database Theory — ICDT'99*, Vol. 1540 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 217–235.

Böhm, J. N., Berens, P. and Kobak, D. (2022). Attraction-repulsion spectrum in neighbor embeddings, *J. Mach. Learn. Res.* **23**(1).

Chari, T. and Pachter, L. (2021). *The Specious Art of Single-Cell Genomics*. **URL:** *https://www.biorxiv.org/content/early/2022/12/22/2021.08.25.457696*

Chuang, C.-Y., Robinson, J., Yen-Chen, L., Torralba, A. and Jegelka, S. (2020). Debiased contrastive learning, *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Curran Associates Inc., Red Hook, NY, USA.

Cox, M. A. A. and Cox, T. F. (2008). Multidimensional scaling, *in* C.-h. Chen, W. Härdle and A. Unwin (eds), *Handbook of Data Visualization*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 315–347.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N. (2020). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, arXiv.

Herrmann, M., Pfisterer, F. and Scheipl, F. (2022). *A geometric framework for outlier detection in high-dimensional data*, arXiv.

Herrmann, M. and Scheipl, F. (2021). A geometric perspective on functional outlier detection, *Stats* **4**(4): 971–1011.

Hinton, G. and Roweis, S. (2002). Stochastic neighbor embedding, *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, MIT Press, Cambridge, MA, USA, pp. 857–864.

Hozumi, Y., Wang, R., Yin, C. and Wei, G.-W. (2021). Umap-assisted k-means clustering of large-scale sars-cov-2 mutation datasets, *Computers in biology and medicine* **131**: 104264.

Kobak, D. and Linderman, G. C. (2021). Initialization is critical for preserving global data structure in both t-sne and umap, *Nature biotechnology* **39**(2): 156–157.

Krumbholz, K., Hardy, A. J. and de Boer, J. (2020). Automated extraction of auditory brainstem response latencies and amplitudes by means of non-linear curve registration, *Computer methods and programs in biomedicine* **196**: 105595.

Lee, J. A. and Verleysen, M. (2007). High-dimensional data, *in* J. A. Lee and M. Verleysen (eds), *Nonlinear Dimensionality Reduction*, Information Science and Statistics, Springer New York, New York, NY, pp. 1–16.

Lee, M. D. (2001). Determining the dimensionality of multidimensional scaling representations for cognitive modeling, *Journal of mathematical psychology* **45**(1): 149–166.

Mardia, K. V. (1978). Some properties of clasical multi-dimesional scaling, *Communications in Statistics - Theory and Methods* **7**(13): 1233–1241.

Marron, J. S., Ramsay, J. O., Sangalli, L. M. and Srivastava, A. (2015). Functional data analysis of amplitude and phase variation, *Statistical Science* **30**(4).
URL: *http://arxiv.org/pdf/1512.03216v1*

McInnes, L., Healy, J. and Melville, J. (2018). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*, arXiv.

Müller, H.-G. (2005). Functional modelling and classification of longitudinal data*, *Scandinavian Journal of Statistics* **32**(2): 223–240.

Ramsay, J. O. (1982). When the data are functions, *Psychometrika* **47**(4): 379–396.

Ramsay, J. O. and Silverman, B. W. (2005). *Functional data analysis*, Springer series in statistics, second edition edn, Springer, New York, NY.

Scheipl, F., Goldsmith, J. and Wrobel, J. (2022). *tidyfun: Tools for Tidy Functional Data*. https://github.com/tidyfun/tidyfun, https://tidyfun.github.io/tidyfun/.

Subbiah, V. (2018). *Pokemon Image Dataset*, Kaggle.

Thalmeier, D., Miller, G., Schneltzer, E., Hurt, A., Hrabe de Angelis, M., Becker, L., Müller, C. L. and Maier, H. (2021). *ABR raw data and results from automated hearing threshold detection*, Zenodo.

Torgerson, W. S. (1952). Multidimensional scaling: I. theory and method, *Psychometrika* **17**(4): 401–419.

van der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms, *J. Mach. Learn. Res.* **15**(1): 3221–3245.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne, *Journal of Machine Learning Research* **9**: 2579–2605.
URL: *http://www.jmlr.org/papers/v9/vandermaaten08a.html*

Wang, Y., Huang, H., Rudin, C. and Shaposhnik, Y. (2021). Understanding how dimension reduction tools work: an empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization, *The Journal of Machine Learning Research* **22**(201): 9129–9201.
URL: *https://dl.acm.org/doi/abs/10.5555/3546258.3546459sec-cit*

Wu, S., Chang, C.-M., Mai, G.-S., Rubenstein, D. R., Yang, C.-M., Huang, Y.-T., Lin, H.-H., Shih, L.-C., Chen, S.-W. and Shen, S.-F. (2019). Artificial intelligence reveals environmental constraints on colour diversity in insects, *Nature communications* **10**(1): 4554.

Xiang, R., Wang, W., Yang, L., Wang, S., Xu, C. and Chen, X. (2021). A comparison for dimensionality reduction methods of single-cell rna-seq data, *Frontiers in genetics* **12**: 646936.

Yang, Y., Sun, H., Zhang, Y., Zhang, T., Gong, J., Wei, Y., Duan, Y.-G., Shu, M., Yang, Y., Di Wu and Di Yu (2021). Dimensionality reduction by umap reinforces sample heterogeneity analysis in bulk transcriptomic data, *Cell reports* **36**(4): 109442.

Zimek, A., Schubert, E. and Kriegel, H.-P. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data, *Statistical Analysis and Data Mining* **5**(5): 363–387.

Declaration of authorship

I hereby declare that the report submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. I am aware that the Thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the report as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future Theses submitted. Further rights of reproduction and usage, however, are not granted here. This paper was not previously presented to another examination board and has not been published.

Munich, 23.01.2024

Name