Topological methods for representational geometries

Shael Brown

Quantitative Life Sciences Program

McGill University, Montreal

February, 2024

A thesis submitted to McGill University in partial fulfillment of the

requirements of the degree of Doctor of Philosophy

Abstract

The branches of systems neuroscience – neurological recordings, computational models and behavior – offer complementary views of human cognition and experience, and can be linked together with a popular analytical framework called *representational similarity analysis* (RSA). In RSA we abstract from measurements and their units to analyze dissimilarities between stimuli in a neural state space. A central assumption of RSA is that matrices of representational dissimilarities, and linear comparison of matrix entries, can capture and compare shape structures, but this assumption remains untested. In this dissertation I introduce a novel framework called *representational topology analysis* (RTA) which can capture and compare shape, i.e. topological, features of representational geometries and provide evidence that RTA can identify, interpret and analyze meaningful features of neural computation that are missed by RSA.

The data structure which RTA uses to capture shape features is called a *persistence diagram* - a form of unstructured data which is not immediately amenable to typical RSA analyses of matrices like machine learning (ML) and inference (IF). In order to solve this problem I have created the first software package, called `TDApplied`, for analyzing persistence diagrams with published methods for ML and IF. By analyzing simulated data and functional magnetic resonance imaging (fMRI) data I demonstrate that `TDApplied` has the potential to identify novel task and behavior-related features of neural function. By comparing `TDApplied`'s efficiency, computational correctness and flexibility against other software packages I also show that `TDApplied` is a valuable tool for applied topological analyses in any domain.

By analyzing data from two vision fMRI studies I then show that RTA, used in conjunction with the specialized tools in `TDApplied`, can segment representational spaces into significant features which are missed by RSA. Therefore, comparisons in RTA can be linked to computational features of neural systems and are more interpretable and trustworthy compared to comparisons in RSA. The first analysis, of data from a famous RSA vision study of IT cortex responses in primates, found a continuous shift in object category in monkey IT cortex which did not exist in human IT cortex. This is a new finding - the original study found evidence of similar functional architectures in human and monkey IT cortex at the level of object category clustering. The second analysis, of data from a naturalistic movie viewing human fMRI study, found that significant region-level representational topological features differed by region, were not captured by representational geometry and may encode stimulus features. It has been found that shared spatial patterns of cortical activity exist across subjects viewing the same naturalistic movie, to different degrees in different brain regions. Similarity of representations suggests similarity of topologies, so with my novel visualization technique for representational spaces called *Proximity Labelled Rips Graphs* (PLRGs) I analyzed three region-level topologies (one early region, one dorsal and one ventral) and found distinct topological signatures which were not accounted for solely by geometry (i.e. RSA) and appeared to capture stimulus features.

Together, these findings situate RTA as a powerful tool for identifying, interpreting and analyzing neurologically-meaningful shape features of representational geometries that RSA cannot detect. Our software `TDApplied` can also be used in other domains and frameworks to carry out efficient and customized analyses of data using shape descriptors, paving the way for new research avenues. Linking neurological data across varied sources with RTA provides new mechanistic insights into neural function across brain, behavior and computational models.

Abrégé

Les modules de la neuroscience des systèmes – l'empreinte de l'activité neuronale, les modèles computationnels et le comportement – offrent des perspectives différentes sur la cognition humaine, et peuvent être liées par le cadre analytique populaire appelé *l'analyse de similarité représentationnelle* (ASR). Avec l'ASR ce ne sont pas les mesures et leurs unités qui sont importantes mais bien les distances entre stimuli dans une représentation d'état neuronal. Un axiome essentiel de l'ASR est que les matrices des distances représentationnelles, et les comparaisons linéaires de matrices à chaque position, peuvent déterminer et comparer les structures de données, mais cet axiome demeure inconfirmé. Dans cette thèse, j'introduis un cadre nouveau, *l'analyse de topologie représentationnelle* (ATR), qui peut déterminer et comparer les structures topologiques dans les géométries représentationnelles. Je démontre que l'ATR peut identifier, interpréter et analyser les structures importantes du codage neuronal qui sont inaccessibles à l'ASR.

L'outil employée par l'ATR pour résumer les structures topologiques est le *diagramme de persistance* – un type de données non-structurées qui ne peut pas être utilisé dans les analyses populaires en l'ASR, comme l'inférence statistique (IF) et l'apprentissage automatique (AA). Pour résoudre ce problème j'ai développé le premier progiciel, `TDApplied`, pour les analyses de diagrammes de persistances avec des méthodes bien-connus d'IF et de l'AA. En analysant des données simulées et des données issues de l'imagerie par résonance magnétique fonctionnelle (IRMf), je montre que `TDApplied` peut identifier de nouvelles caractéristiques du codage neuronal associées aux tâches et aux comportements. À l'aide des analyses comparatives sur

la vitesse, la flexibilité et les erreurs de calcul de `TDApplied` et d'autres progiciels, je fournis la preuve que `TDApplied` est un outil efficace pour l'analyse topologique dans tout domaine.

En analysant les données de deux études IRMf sur la vision, je montre que l'ATR, combiné avec `TDApplied`, peut segmenter les géométries représentationnelles en structures topologiques significatives qui sont inaccessibles à l'ASR. Donc les comparaisons de l'ATR peuvent être liées avec les caractéristiques computationnelles des systèmes et sont plus fiables et interprétables que celles de l'ASR. Dans ma première analyse, avec les données d'une célèbre étude ASR sur la vision concernant les réponses du cortex temporal inférieur primate, j'ai découvert dans le cortex singe une progression graduelle pour la catégorisation d'objets qui n'existe pas dans le cortex humain. Il s'agit d'une découverte notoire, car l'étude originelle a trouvé des architectures fonctionnelles similaires entre les deux cortex au niveau du regroupement de catégories d'objets. Pour ma deuxième étude, j'ai analysé les données d'une étude IRMf qui a trouvé des motifs d'activité neuronale communs, qui variaient par région, entre sujets visionnant le même film naturel. Une similarité d'activité suggère une similarité des topologies et donc avec l'ATR et ma nouvelle méthode de visualisation des espaces représentationnels le « Proximity Labeled Rips Graphs » (PLRG) j'ai pu différencier trois topologies locales (une région primaire, une dorsale et une ventrale). De plus, ces topologies n'étaient pas capturées par l'ASR et semblaient capturer des caractéristiques des stimuli.

En somme, ces résultats indiquent que l'ATR est un outil puissant pour identifier, interpréter et analyser les structures topologiques représentationnelles qui font partie du codage neuronal et qui sont indétectable à l'ASR. Notre progiciel `TDApplied` peut aussi analyser les données dans d'autres domaines pour ouvrir la voie à des recherches futures. En liant les différentes données neurologiques avec l'ATR, nous pouvons accéder à de nouvelles informations mécanistiques sur la fonction neuronale à travers le cerveau, le comportement et l'informatique.

Acknowledgements

My completion of this thesis would not have been possible without the financial, academic and personal support I have received from numerous sources.

I will be forever grateful for all the support, mentorship and understanding I have received from my advisor, Reza. This thesis was born out of one of our first mad-scientist brainstorming sessions (despite my initial stubbornness), and these sessions were some of the most fun experiences of my degree. You have calmed me down when I was stressed and re-motivated me when I felt stuck. You challenged me and pushed me to grow as a scientist, and the skills of argumentation, which I initially thought were uninteresting and tedious, I now consider to be one of my greatest assets. You have given me your time and effort, and it has been an honour and a pleasure learning from, and scheming with you. I wish you nothing but continued success and happiness in the future, and most importantly a flourishing TDA program in the lab.

I have also received valuable feedback on my projects from my committee members and comprehensive examiners, Professors Arjun Krishnaswamy, Bratislav Misic, Peter Savadjiev, Erik Cook and Boris Bernhardt, and for this feedback I am extremely grateful.

To my QLS support team, Alex DeGuise and Professors Celia Greenwood and Mathieu Blanchette, I sincerely thank you for answering all of my many questions and encouraging me throughout this whole process. I feel extremely fortunate to have had such a caring program administrative team, and please know that you all have had a significantly positive impact on my PhD experience.

I am incredibly fortunate to have worked alongside incredibly intelligent, talented, fun and kind lab-mates, and I thank you all for making this experience easier and more enjoyable.

To my dad, pa. You are the reason why I've taken this path, and I feel very blessed to be on it. You have brainstormed with me, even when I was stubbornly insisting that I was right, and you have edited my projects dozens of times on a moment's notice. You have helped me navigate academia, saving me from countless mistakes, and have kept me motivated when I was feeling down. Thank you for being my north star throughout this process, I love you so much.

Thank you to my whole family for supporting me before and during my degree. Zeebs, you have made me laugh when I wanted to cry, and cry when I wanted to laugh (in a good way). Mum, you have been and will always be my sounding board and confidante. Joe and Bev, you guys have made Montreal feel like home for me, fed me and kept me laughing non-stop. I love you all.

Next, I want to thank my friends Bill and Bob for their loving support these past few years. Couldn't have made it without you!

Finally I would like to thank my wonderful, supportive and amazing wife, Nicole. Bubs, we made it! Thank you for everything you've done while I've walked this path, and all the support you will continue to give me in the future. This accomplishment is ours to share together, and G-d willing we will have many more to celebrate in the future. Thank you for everything, my love.

For all the people in my life who loved and supported me during this process, this is for you.

Funding

The projects in this thesis received funding from the 2016 CIHR grant for cortical mechanisms of 3-D scene and object recognition in the primate brain and from the department of Quantitative Life Sciences at McGill University.

Contributions

My doctoral thesis introduces a novel framework, RTA, for capturing and comparing shape features of representational geometries, by combining the popular tools of representational similarity analysis and persistent homology. My approach is distinct to RSA by replacing representational dissimilarity matrices (RDMs) and the correlation/correlation distance between them with shape descriptors called persistence diagrams and tools from the mathematical literature called the persistence Fisher kernel/wasserstein and bottleneck metrics. My approach is also different from other RSA variants which model non-linear dependencies between RDM entries by capturing dependencies between any number of representational dissimilarities. The toolbox I developed, `TDApplied`, is the first publicly available software for analyzing persistence diagrams with published methods for statistical inference and machine learning in either R or Python.

Through three analyses of neurological data I tested the hypothesis that RTA captures representational differences which RSA misses. My results supported this hypothesis, demonstrating that shape features of representational geometries can be used to interpret representational differences via stimulus features and behavior. Finally, I tested the hypotheses that my toolbox is the fastest, most flexible and most computational sound R package for topological analyses. My results supported this hypothesis, positioning `TDApplied` as a powerful tool for topological analyses of data.

This doctoral thesis was prepared in accordance with McGill's Faculty of Graduate and Postdoctoral Studies manuscript-based thesis guidelines. Chapter 2 has been submitted to the Journal of Open Source Software (JOSS) at the time of initial submission and the software documentation is currently available at the public repository for R packages (CRAN), and Chapter 3 has been submitted to the Proceedings of the National Academy of Science (PNAS).

Contributions of Authors

Chapter 2: `TDApplied`: An R package for Machine Learning and Inference with Persistence Diagrams

Shael Brown and Reza Farivar

I conceived of the project and wrote the code, documentation and tests. RF provided guidance on functions that should be included in the software, feedback on package usability and editing on package documentation.

Chapter 3: The Topology of Representational Geometry

Shael Brown and Reza Farivar

RF and I conceived of the project, and I carried out the analysis with the advice and guidance of RF. I wrote the manuscript and designed all figures, with editing from RF.

List of Abbreviations

RSA	representational similarity analysis
RDM	representational dissimilarity matrix
RSM	representational similarity matrix
TDA	topological data analysis
PH	persistent homology
RTA	representational topology analysis
V1/V3	visual area 1/3
fMRI	functional magnetic resonance imaging
PHC1/PHC2	parahippocampal area 1/2
LO1/LO2	lateral occipital area 1/2
V3a/V3b	visual region 3 a/b
HCP	Human Connectome Project
GLM	generalized linear model
MDS	multidimensional scaling
PCA	principal component analysis
kPCA	kernel principal component analysis
SVM	support vector machine

List of Figures

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Table of Contents

Chapter 1

Literature Overview

Representational similarity analysis (RSA, Kriegeskorte, Mur, and Bandettini, 2008) is the first neuroimaging analysis framework in which data can be compared across species and imaging modalities (Kriegeskorte, Mur, Ruff, et al., 2008; Kriegeskorte, 2009), and against computational models and behavior data (Kriegeskorte, Mur, Ruff, et al., 2008; Rothlein and Rapp, 2014; Tamir et al., 2016; Wasserman et al., 2017). These types of comparisons are possible in RSA by abstracting measurement units in the two data sources (for example fMRI voxel activity, neuron binned spike counts, etc.) to stimulus-stimulus dissimilarity values stored in *representational dissimilarity matrices* (RDMs), and the function used to compute these dissimilarities is called a *first-order isomorphism*. We can then compare pairs of corresponding values in two RDMs with a *second-order isomorphism* function – the Spearman correlation is often used due to its simplicity and scale-invariance property. In order to capture more expressive relationships between RDMs beyond correlation, which can only capture linear relationships, several studies have suggested using other second-order isomorphisms for comparing RDMs (Diedrichsen et al., 2020; Edelman, 1998; Kriegeskorte and Kievit, 2013; Shahbazi et al., 2021).

Despite the broad palette of second-order isomorphisms for many experimental scenarios, all options to date compare corresponding entries of RDMs in order to quantify similarity or difference of representational geometries. Each RDM entry is a represen-

tational dissimilarity between two conditions, so comparing only pairs of these values obscures any similarities or differences in the representational geometries that are comprised of more than two dissimilarities and conditions. But the central assumption of RSA is that an RDM defines a geometry, or shape, and shape structure is best appreciated globally, i.e. viewing all first-order comparisons simultaneously. To see why shape structures are best appreciated globally, let us consider two examples from the literature which use mathematical modeling based on biological assumptions to identify global shape structures in representational geometries. First, it has been shown that the representational geometry of orientation-selective neurons in primary visual cortex is a loop – smoothly-varying responses that repeat every 180 degrees (Singh et al., 2008). Second, the representational geometry of rat grid cells, responsible for helping the rat determine its location in space, is a torus (i.e. hollow doughnut) (Curto, 2017). Viewing only a small segment of either shape would not reveal its global properties, i.e. a segment of a loop would not capture its periodic nature.

These two representational geometries, a loop and a torus, contain shape features which can only be appreciated at a global scale by considering distances between all points (i.e. representations) simultaneously – the former contains a single loop, and the latter contains two different types of loops (one which bisects the doughnut horizontally and the other which bisects the hollow tube vertically) and one void inside the tube. These features also vary in dimension – a loop is 1-dimensional and a void is 2-dimensional, like the casing of a sphere. Under this shape-of-data perspective, the two representational geometries are distinct because they contain different numbers of shape features, and their features are of different dimensions. And we know that orientation-selective cells and rat grid cells perform distinct calculations, but in Figure 1.1 I provide a visual illustration as to how RSA could erroneously equate the two representational geometries, of a torus and a loop. In this illustration I show how representations which exist on a torus can be projected onto projections within an annulus (i.e. a loop with added noise), and the

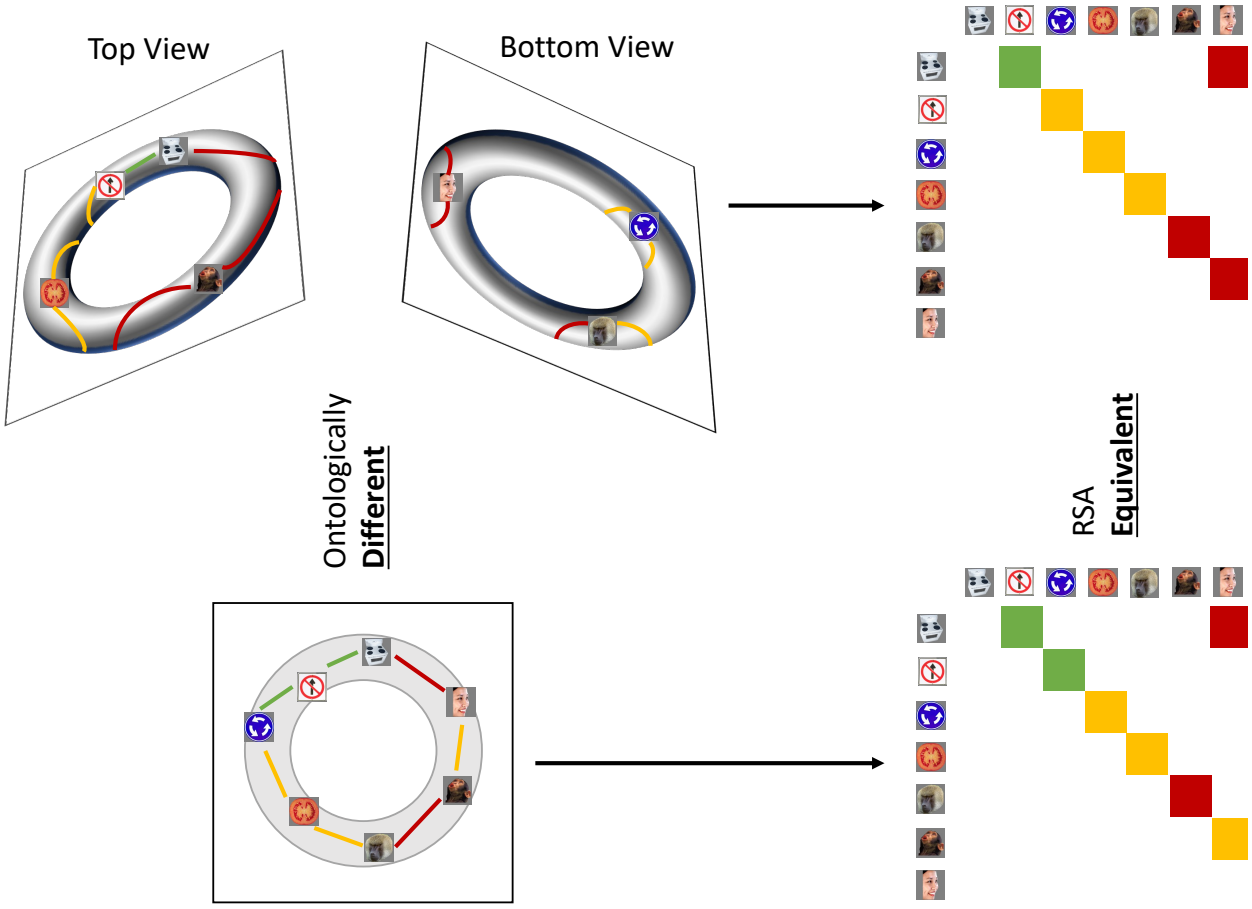

Figure 1.1: Sampled representations from a torus, projected onto an annulus, and the resulting two RDMs. The stimuli images were obtained from the supplemental information in Kriegeskorte, 2009 but were originally introduced in Kiani et al., 2007, as is the case in later figures. Top left are the top and bottom views of seven sampled representations from the surface of the torus, with colored lines indicating representational distances between adjacent points (green for small distances, yellow for medium and red for large). Bottom left is the projection of these torus representations onto an annulus, with updated representational distances. These distances for both shapes are color-coded in their respective RDMs, which would be considered equivalent by RSA, despite the representational spaces having completely different shapes. This figure is the same as Figure 3.1.

two RDMs would be erroneously equated by RSA despite being generated from different shapes. I will refer to this illustration as our RSA "counterexample".

Previous studies have demonstrated that RSA can erroneously find similarity in two computational systems (for example two artificial neural networks) that are performing entirely distinct calculations (Chen et al., 2021; Dujmović et al., 2022). In Chen et al., 2021 the error came from (1) using biologically implausible feed-forward model architectures of the human reading system (a system that is is known to utilize feedback mechanisms of orthographic processing), and in Dujmović et al., 2022 the error arose from (2) confounding variables which were correlated with stimulus condition, and (3) computing representational dissimilarity using different functions which project representations onto distinct shape structures for comparison. Our RSA counterexample of the torus and loop representational geometries is similar to (1) because we know that the computational model of an orientation-selective cell, a loop, would not be a biologically appropriate model of a rat grid cell, a torus, and vice versa. On the other hand, in our RSA counterexample we were able to identify the difference between the two representational spaces based on their shape features, whereas the difference could only be identified based on underlying assumptions about the biology of the human reading system and is therefore not data-driven. Our RSA counterexample is also similar to (2) in that the difference between the two representational spaces was due to differences in the features which define the spaces. However, (2) was illustrated with two high-performing convolutional neural networks, trained to recognize ten classes of naturalistic images, having highly similar final layer representations (despite distinct computational mechanisms). Since the models were high-performing it would be expected that the two final-layer sets of image representations were well-clustered according to image class. On the other hand, we would expect that earlier-layer representational geometries of the two neural networks would differ, with the network trained with the confounding dataset displaying more clustered representations and not so for the other network – differences that could be detected using shape analysis. Lastly, in (3) the two first-order isomorphisms constrain the representations to live on distinct shape structures – only one of which contains a void. While specific samplings of the two embedding spaces could have similar geometry by chance

(as was shown in Dujmović et al., 2022), sufficient sampling of the space that contains the void would capture the void structure (i.e. with points on all sides of the void) while sufficient sampling of the space that does not contain the void would not capture any significant void structure. Therefore, the difference between the two spaces would have been detectable using shape analysis.

In order to resolve our new criticism of RSA, illustrated by our counterexample in Figure 1.1, in Chapter 3 I will introduce a novel framework, called representational topology analysis (RTA), which can capture and compare various types of shape features in representational geometries. In this framework an algorithm called *persistent homology* will be used to calculate a shape descriptor, called a *persistence diagram*, of representational geometries defined by RDMs which contains information about shape features like clusters, loops and voids. I will demonstrate that RTA can identify significant shape features in representational spaces which RSA cannot, that these features can be used to find a previously unseen functional difference of human and monkey IT cortex in an object-viewing study and that this difference is explainable and mechanistic. Moreover, I will show that these shape features vary across regions in a naturalistic movie-viewing study and likely represent stimulus features.

A framework which satisfies the characteristics in the previous paragraph, however, would not be a suitable counterpart to RSA without the ability to carry out typical RSA analyses. Two examples of such analyses include multidimensional scaling (MDS) (Mead, 1992), in which multiple RDMs and their pairwise (Spearman linear) distances are used to define a low-dimensional embedding which captures the inter-RDM relationships, and model inference (Kriegeskorte, Mur, and Bandettini, 2008), in which (groups of) RDMs can be compared statistically for similarity or difference. These two analyses are examples of machine learning and statistical inference respectively, two extremely popular approaches for analyzing data. While machine learning and statistical inference procedures tailored for the unstructured data in persistence diagrams have been published (Fasy et al., 2014; Le and Yamada, 2018; Robinson and Turner, 2017), no publicly-available soft-

ware package in R or Python has offered these functionalities. In Chapter 2 I will introduce a novel R software package I have authored, called `TDApplied`, for analyzing persistence diagrams with machine learning and statistical inference. Through a variety of simulations I demonstrate that `TDApplied` is a more efficient, computationally sound and flexible package than other R packages for topological analyses of data, and by analyzing neurological data from HCP I show that `TDApplied` can extract meaningful insights from (neurological) data in ways that other packages cannot.

Viewed as a whole, RTA with `TDApplied` is a powerful framework for shape analysis of representational geometries which can capture meaningful, and otherwise hidden features of neural function and analyze these features in typical RSA workflows.

1.1 Representational similarity analysis

1.1.1 RSA origins

There are three dominant and complementary modules in systems neuroscience – neural activity measurement, computational modeling and behavior (Kriegeskorte, Mur, and Bandettini, 2008) – and combining data (and therefore inferences) between the modules is an enticing prospect. The statistical power of a study can be increased when pooling multiple data sources – an obvious example would be across subjects, but we could also pool data across various neuroimaging modalities (for example, MRI) with appropriate methodologies (Calhoun and Sui, 2016; Glasser et al., 2016; Huster et al., 2013). Pooling data across the different modules would also allow researchers to test novel hypotheses which are comprised of the relationships between brain, model and behavior. Unfortunately, there are a number of challenges facing the pooling of data between (and even within) the modules. Behavior data, like perceptual similarity judgements in psychophysics, is the result of neural computations but is only linked to those computations via hidden and complex encoding and decoding mechanisms of stimuli and neural features (Kriegeskorte and Diedrichsen, 2019; Laakso, 2000). On the other hand, brains can

vary significantly in their anatomy and functional wiring, and artificial neural networks can vary significantly in their model architecture (for example different numbers of layers and hidden units), making comparisons between brain and model (and even between brains and between models) very challenging (Laakso, 2000). RSA is a simple framework which solves this problem of data comparison across and within modules in three steps: (1) computing representations of stimuli, (2) comparing representations with representational (dis)similarities, storing the results in a *representational (dis)similarity matrix* (RDM/RSM) which encodes the space's *representational geometry* and (3) comparing representational dissimilarities using a *second-order isomorphism*. For certain types of data (such as perceptual similarity judgements and perceptual models) it is possible to even skip steps (1) and (2) to directly obtain an RSM/RDM without needing to capture any notion of representation. In the language of RSA, step (2) can be viewed as calculating *first-order isomorphisms* between representations.

The key to RSA's success, hidden within the simplicity of each of its three steps, is a number of deep ideas from psychology. In step 1 of RSA representations are computed, but exactly do we mean by a representation? In the case of brains or neural models, the *theory of pattern activations* defines a *representation* as an activity profile across functional units, i.e. a snapshot of what the brain is doing in a single instance (Churchland, 1995), which occupies a position in a neural state/activation space (Churchland, 1986). This definition is intuitive and idealistic, but may present issues in practice – in fMRI studies spatial-temporal noise (T. Liu, 2016) may obscure signal in individual time point activity patterns. In such studies representations could also be defined as generalized linear model coefficients across voxels (Connolly et al., 2012; Hendriks et al., 2017), but this approach would obscure temporal information via averaging. The pattern activation theory definition of representations should be suitable in fMRI RSA studies, so long as sufficient steps are taken to reduce potential noise sources via preprocessing, for example spatial smoothing or more advanced denoising techniques (Jo et al., 2010; T. Liu, 2016).

The second step of RSA compares representations, and in the pattern activation theory such comparisons are meaningful because the proximity of representation positions in state space determines perceptual similarity/dissimilarity (Churchland, 1986), establishing a "parallelism" between brain/model and behavior (Shepard and Chipman, 1970). Perceptual dissimilarity, the principal object of study in the field of pscyhophysics, is based on the idea that the brain featurizes the external world into a state space in which (dis)similarity judgements are are simply the proximity between pairs of points, and that the dimensions of this space need not be entirely describable (Goodman, 1951; Shepard and Chipman, 1970). Small perturbations of an object's representation in a state space, as long as the perturbation does not affect the relative position of the representation compared to other object representations, should not change the qualitative experience of those objects (Laakso, 2000).

One problem with these ideas is that it is unclear how to select appropriate functions to compute representational (dis)similarity. For example, human judgements are not symmetric, so it has been argued that neither should representational dissimilarity functions (precluding simple and popular methods like Euclidean distance and cosine distance) (Edelman, 1998). Even if a neural activity space contained enough neurologically-relevant dimensions to completely reflect stimulus encoding, it is unlikely that any simple vector metric would be a biologically-plausible judgement mechanism. In Shepard and Chipman, 1970 it is also argued that a parallelism exists between neural representations and perception of stimulus features (i.e. behavior), implying that future studies should attempt to learn more biologically-plausible neural (possibly non-linear) representational dissimilarity functions based on behavioral dissimilarity judgements. The study Bobadilla-Suarez et al., 2019 compared a number of dissimilarity functions of fMRI spatial patterns, where each function was a well-known mathematical function of vectors, to see which function's stimulus-stimulus distance matrix best approximated the confusion matrix of a well-performing classifier (decoding the stimulus being viewed at each time point based on the time point's spatial activity pattern). Their results indicated that

certain functions performed better than others; however, decoding models impose assumptions on neurological data which may not be neurologically plausible and therefore comparing against behavioral similarity judgements would be more appropriate. Another study, Yousefnezhad et al., 2021, used neural networks to learn flexible non-linear dissimilarity functions between representations to define RDMs which are more appropriate for comparison using typical linear second-order isomorphisms like correlation, but such representational comparisons are not interpretable/explainable and are driven by computational assumptions alone (such as modeling neural responses using the fMRI experiment design matrix in the GLM framework and that the brain computes gradients in order to learn representations) as opposed to biological assumptions or a mix of the two.

The third step of RSA is the comparison, with correlation, of two sets of corresponding representational dissimilarities, an idea first proposed for behavior data in Shepard and Chipman, 1970 and later being proposed for neural networks in Laakso, 2000. Distinct systems may perform (nearly) identical calculations, for example two people with different sized brains may correctly solve an addition problem following the exact same steps, but the neural state spaces of the two subjects would be different if the two brains had different numbers of neurons so individual representations would not be comparable. Therefore, what is more interesting than individual representations is the relative positioning between multiple representations. Collections of representational distances define a ”shape”, which is invariant under rotations and translations. However, shapes should be identifiable as long as *enough* points are sampled from them, but in the RSA framework it is only possible to compare representational spaces with the same number of points (correlation only operates on two same-sized vectors). It was also remarked that it would be interesting to compute matches between representations of stimuli from different domains, although the originally proposed framework did not seem entirely appropriate for doing so (Laakso, 2000).

In both Shepard and Chipman, 1970 and Laakso, 2000, Pearson correlation is used to compare representational distances, largely because it is scale invariant – unaffected by the mean and variance of its two input vectors. This property is desirable because representations, and therefore their dissimilarities, may exist on separate scales in two systems even if they encode the same computations. On the other hand, any function comparing two sets of (corresponding) representational dissimilarities can be a second-order isomorphism. In Kriegeskorte and Kievit, 2013 it was suggested that non-linear second-order isomorphisms (as opposed to linear ones like correlation) may be useful for capturing non-linear features of representational geometries, and several studies have proposed isomorphisms along these lines. For example, Diedrichsen et al., 2020 proposed a metric of independence between representational distances called distance correlation (Szekely et al., 2008), which can capture linear and non-linear relationships while also reducing bias in RSA experiments. Other approaches have calculated distances between correlation matrices (commonly used as RSMs in RSA) based on the high-dimensional structure (called a "manifold") on which all correlation matrices exist (Shahbazi et al., 2021; You and Park, 2022) – correlation matrices form a subset of all matrices with special structure. Despite the performance gains exhibited by these non-linear and more complex second-order isormophisms, they each have drawbacks. Distance correlation is only able to model non-linear dependencies in the joint distribution of the two sets of representational dissimilarities, whereas there may exist shape structures in the two spaces (like was discussed in Laakso, 2000) which inform similarities/differences. On the other hand, the distribution of correlations that arise specifically in neuroimaging data would likely form a subset of the correlation-manifold (i.e. a "sub-manifold" of all possible correlation matrices), meaning that we may be greatly overestimating distances. Large distances are the least informative when determining shape structure because connections exist only locally between neighboring points on the shape (i.e. there are no discontinuous jumps on a shape, only between clusters).

All of these ideas together contributed to the development of RSA. Neural activity representations of stimuli give rise to representational geometries (which define shapes in representational state space), regardless of whether the representations were measured in brains or computational models. Geometries can also be computed from behavior data, like perceptual judgements in psychophysics. Any of these geometries can then be compared using a second-order isormorphism, comparing computations and experiences regardless of functional wiring/architecture. By comparing behavior data with neural data we implicitly assume the stimulus differences are parallel with neural state differences, and by utilizing RSA to bridge the three modules of systems neuroscience we recruit tools from decades of psychological theory.

1.1.2 RSA successes in vision fMRI studies

One domain in which RSA has had a significant impact is vision fMRI, which was the application of RSA's introductory paper (Kriegeskorte, Mur, and Bandettini, 2008); fMRI recordings of human IT cortex during object viewing were collected and RDMs were compared against computational-model RDMs. Subsequent studies (Kriegeskorte, Mur, Ruff, et al., 2008; Kriegeskorte, 2009) compared, using RSA, the human fMRI data against electrode recording data (Kiani et al., 2007) from monkeys viewing the same stimuli, identifying similar functional architectures between the two species. These were landmark studies, providing poignant evidence that (1) monkey IT cortex is a suitable model for human IT cortex for object recognition and (2) RSA is able to test hypotheses that other methods cannot. RSA lends itself nicely to vision fMRI applications because (collections of) neurons in visual areas compute visual features within their field of view, with simpler representations in earlier areas (such as orientation-selective neurons in V1) and more complex representations in later ones (such as face cells in IT cortex), and these feature maps are detectable in vivo in humans using fMRI (Kamitani and Tong, 2005; Kanwisher et al., 1997; Mack et al., 2013). Since primates have similar functional architectures for vision (Mack et al., 2013; Nassi and Callaway, 2009), and invasive techniques like electrode

recordings can record neural activity in monkeys at a high spatial resolution, inter-species and inter-modality RSA could boost statistical power for inferences made about human vision. For reasons such as these, RSA has been utilized in many insightful fMRI vision studies.

While inter-species and inter-modality studies are interesting applications of RSA, one of the most significant contributions of RSA to vision research was performed completely within human subjects using only fMRI. The ventral and dorsal visual streams have been known to process, with significant interactions, object identity and object location/motion respectively (Mack et al., 2013; Nassi and Callaway, 2009; Ungerleider and Mishkin, 1982), but these interactions have yet to be well-characterized. In Bracci and Op de Beeck, 2016, object representations in human ventral and dorsal areas were disentangled according to object shape and category using RSA, comparing against perceptual similarity judgements of the objects. It was found that representational spaces changed continuously from category-selective in later ventral areas, to mixed areas, to shape-selective later dorsal areas, suggesting that later visual areas in both streams are more specialized whereas early and middle areas have significant overlap. These results are a significant achievement, which would not have been possible without RSA, as stimulus representations in different brain regions with different anatomical shapes cannot be directly compared.

The ability of RSA to compare fMRI data across regions of different shape has led to its application in aging studies (the "same" brain region in adults and children would be of different sizes). For example, in Golarai et al., 2017 and Cohen et al., 2019, visual stimuli (images of faces and objects from different categories) were shown to a group of children and a group of adults during fMRI scanning, and in neither paper was a difference found between the child and adult representational spaces in IT cortex. The authors of these two studies concluded that object representations are likely fixed at an early age, however it is possible that their methods were insufficient to capture more complex non-linear group differences. One interesting feature of both studies is how the two groups (child and adult) were compared. In Cohen et al., 2019 each group had a mean RDM computed

and the two mean RDMs were compared with a permutation test, which is the suggested approach from Kriegeskorte, Mur, and Bandettini, 2008. However, in Golarai et al., 2017 a permutation test of within-group correlations was employed thereby capitalizing on within-group variations rather than averaging over them. This latter approach to group inference in RSA has a direct counterpart in RTA, which we will discuss in Chapter 2.

While the previously mentioned studies have focused solely on vision at a regional level, the flexibility of RSA can be used to compare different visual representations of stimuli and at a fine scale. An example of such a study is Devereux et al., 2013, in which the pictorial and semantic representations of words were compared using a local searchlight-based approach (Kriegeskorte et al., 2006) combined with RSA. Their results indicated that clusters of local-scale representational spaces exist which are representation-agnostic, implying that cross-representation studies could result in greater inferential power. Of particular interest in this approach is that RSA is performed even when the stimuli are not the same, in contrast to the previously listed studies. Despite the obvious correspondence between the cross-modality stimuli, like viewing a word or a picture representing that word, which provides a sensible roadmap for the application of RSA, this study is a basic example of the cross-domain experiments which were hypothesized to be interesting applications of RSA in Laakso, 2000. Therefore, it remains an open question to what extent different stimuli can be used in a single RSA study.

The unique value of RSA in all the studies described above is comparing data which was not comparable using other methods in order to make meaningful inferences about vision. Whether the comparisons were between species, modalities, brain regions, age groups or stimulus representations, the raw activation patterns of stimuli did not exist in a shared representational space, but their representational dissimilarities were comparable. By leveling the playing field across these comparisons we can potentially use a number of data sources (ones which perhaps take less time or money to collect compared to fMRI) to make inferences about human vision with increased sample size, and this is an idea

which is briefly discussed in Chapter 3. In summary, RSA is a powerful and flexible tool for studying human vision with fMRI.

1.1.3 RSA criticisms

Despite the widespread adoption and success of RSA, especially in the case of vision fMRI studies, there have been several major sources of criticism. The first criticism has already been discussed at the beginning of Chapter 1 – high correlation of RDMs, i.e. second-order isomorphism values, does not imply similar calculations being carried out (Chen et al., 2021; Dujmović et al., 2022) – which is perhaps the most problematic issue with RSA (and is also a major challenge in model adjudication across many scientific fields). However, other issues arise when calculating second-order and first-order isomorphisms due to fMRI-specific and general RSA biases.

In Cai et al., 2019 and Viviani, 2021, mathematical derivations demonstrate how, in the GLM framework, non-independence of fMRI design-matrix columns as well as spatio-temporal autocorrelation biases RSA inferences made from second-order isomorphism calculations. Simulations also showed that these effects can cause inflated correlations between task RSMs and resting-state RSMs or even with RSMs generated from random noise, whether the RSMs were computed by comparing GLM coefficients with Euclidean distance or correlation distance (i.e. one subtract correlation), or by correlating spatial patterns. On the other hand, second-order isomorphisms can be inflated/defalted due to non-independence between first-order isomorphism values, for example those which share an underlying representation (Diedrichsen et al., 2020). First-order isomorphisms, like Euclidean and correlation distance, also can be inflated by measurement noise especially for small (close to 0) distance values, an inflation which would bias second-order isomorphisms when the measurement noise is non-uniform across representations (Diedrichsen et al., 2020). The solutions proposed by these studies suggest that unbiased (cross-validated) first-order isomorphisms combined with incorporating the covariance (dependence) structures within RDMs into second-order isomorphism calculations

would provide more robust RSA inferences in fMRI studies. On the other hand, from a preprocessing perspective, it has been shown that certain methods for denoising fMRI data can improve RSA inferences (Charest et al., 2018; Hendriks et al., 2017; Prince et al., 2022). The current literature therefore points towards a careful combination of preprocessing, postprocessing and interpretive techniques for meaningful applications of RSA to fMRI studies. However, it seems that most RSA studies still utilize the original, simpler framework.

In Chapter 3 we use RTA to capture and compare shape structures in spatial-pattern correlation-distance RDMs from fMRI data, so what would be the effect of these RSA biases on shapes and their comparisons? While outside the scope of this thesis, it is possible that effects of bias would be lower for several reasons. Firstly, higher-dimensional shape structures, like loops or voids, are defined by specific patterns of non-independence (i.e. autocorrelation) between distance values, meaning that shape detection may actually benefit from non-trivial RDM covariance matrices. Secondly, we explained at the beginning of this chapter that shape analysis can detect differences in dimension between highly-similar (measured by correlation) representational geometries, and therefore more topologically trivial randomly-generated fMRI data may be more distinguishable to real, more topologically rich fMRI data (we show in Chapter 2 that task-fMRI data can contain loop structures of correlated spatial patterns). In Chapter 1.2.2 we will see that data shapes are robust to small amounts of additive noise and only depend on the ordering of representational dissimilarity values rather than the values themselves, suggesting that uniform measurement noise or low variance non-uniform noise would not affect shape calculations or comparisons.

1.2 Topological data analysis

In order to capture the ”shape of data” we will utilize techniques developed from the mathematical area which is concerned with shape structure – *topology*. In the topological

approach, broadly speaking a shape (i.e. a topological space) is defined by a set of points and a collection of neighborhood subsets, which determine the local-scale connections between adjacent points. Global shape structure can then be described using a technique called *homology*, a useful tool for determining when two shapes are distinct. Example applications in neuroscience will be provided to give an intuitive introduction and motivation for the use of topology and homology.

1.2.1 Topology and homology

One of the most popular frameworks for analyzing neuroimaging data is also one of the simplest examples of a topological space – networks (called graphs in the mathematical literature). For example, functional connectomes are networks which capture the temporal coherence of activity between functional units, and these networks can be used to distinguish between subjects with certain neurological conditions and controls (Y. Liu et al., 2008; Supekar et al., 2008; Wang et al., 2009). Mathematically, a network is a pair (V, E) where V is a set of objects (the "nodes" or "vertices", such as voxels in an fMRI dataset) and E is a collection of pairs of those objects (the edges between pairs of nodes, such as significant correlations between the temporal activity of voxels). A network can be either weighted or unweighted, depending on whether we assign a numeric value (i.e. weight) to each edge (e.g. a voxel correlation value) or not, and can be either directed or undirected, whether edges are directed (encoding asymmetric causal relationships) or not (encoding symmetric relationships). For any network node v, the set of all nodes which v has an edge with is called the 'neighborhood' of v (for example, all voxels which have a significant correlation to a target voxel).

Topological spaces can be thought of as generalizations of networks: a topological space is a pair (X, T) where X is a set of objects (like voxels) and T is a special collection of "open" subsets of those objects, allowing for the relationships (i.e. interaction) between any number of the objects in X (as opposed to only permitting interactions between two vertices in a brain network). This relaxed property is more appropriate for neurologi-

cal data because functional dependencies can exist between multiple (i.e. greater than two) functional units (Sizemore et al., 2019). Topology captures nearness, i.e. adjacency, between collections of objects in a space.

A very useful type of topological space is called a Δ-*complex* (Hatcher, 2002), often called a "(simplicial) complex" in the applied topology literature. The latter terminology is the one we will use from this point onwards. In a complex we glue together multiple simple shapes in a special way. The building blocks of these spaces are called "n-simplices", of dimension n. Examples are vertices (0-simplices), edges between pairs of vertices (1-simplices), triangles between triples of vertices (2-simplices) etc., and all simplices exist in some Euclidean space (a data space of fixed dimension k in which the distance between any two points in the space is the Euclidean distance). The dimension n of each n-simplex $[v_0, v_1, \ldots, v_n]$ is the number of parameters needed to locate a point on that shape. In a complex, n-simplices are glued (i.e. intersect) along lower-dimensional simplices, for instance two triangles could be glued along one (or more) edges/vertices. There is a straightforward relationship between complexes and networks – the set of 1-simplices of a complex forms a network called the complexe's "1-skeleton". In Figure 1.2 we can see an example of a complex formed by the gluing of several triangles, edges and vertices in three separate connected components.

Complexes are shape structures (possibly high dimensional and non-visualizeable) which require special tools to identify their essential structure. In the simpler case of networks, various well-known "graph metrics" can be computed summarizing different aspects of a network's structure, such as how segregated the graph is into modules or how integrated the graph is (Rubinov and Sporns, 2010). While a number of graph metrics have been extended to analyze complexes (Giusti et al., 2016), there is one particularly useful framework for determining the shape structure of complexes for comparison -– *homology* (Hatcher, 2002). Homology detects the number of distinct (i.e. independent) holes, of various dimensions, that exist in a complex, and it turns out that if two complexes have different homology then they are not the same shape. A

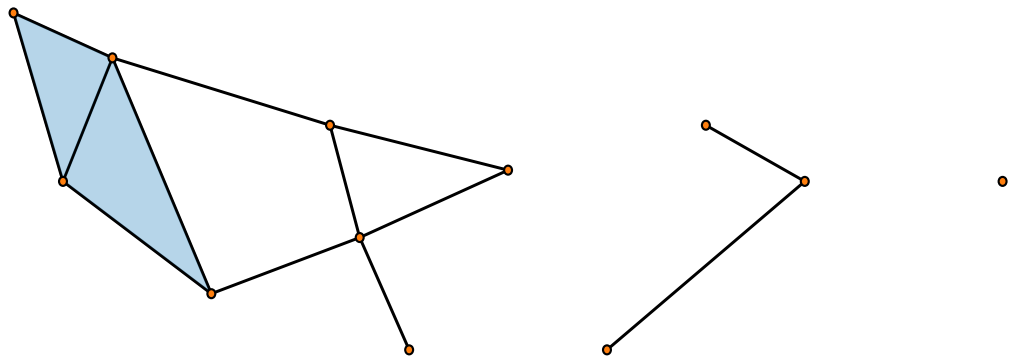

Figure 1.2: A sample complex with three connected components. The left-most component contains 2-simplices, 1-simplices and 0-simplices, the middle component contains 1-simplices and 0-simplices, and the right component contains just a single 0-simplex.

hole has a dimension, which is the number of parameters which encode the position of a point around the hole, and some examples of holes are clusters (0-dimensional), loops (1-dimensional) and voids (2-dimensional). This may seem like a strange approach to data analysis – normally we try to ascertain where our data *is* by modeling distributions, performing clustering, etc. But by finding holes in our data we are actually able to answer the dual question of where our data *is not*, with added information about the local dimension of our data (i.e. how many latent variables adequately capture the variance of each data segment). Interestingly, it has been suggested that holes in neural activity space may represent parallel processing strategies, via divergence and later re-convergence of neural activity (Sizemore et al., 2019).

In order to compute the homology of a complex we need a piece of machinery called the *boundary map*, ∂. The boundary of an n-simplex $[v_0, v_1, \ldots, v_n]$ is the sum of all its $(n-1)$-simplices, resulting in a lower-dimensional complex – for example the boundary of a 1-simplex $[v_0, v_1]$ is $\partial([v_0, v_1]) = v_0 + v_1$ and the boundary of a 2-simplex $[v_0, v_1, v_2]$ is $\partial([v_0, v_1, v_2]) = [v_0, v_1] + [v_1, v_2] + [v_2, v_0]$. This definition of boundary implicitly assumes that the ordering of vertices in an n-simplex is irrelevant (and all additions are modulo

2) as would be the case for functional connectivity correlation connections between brain regions. By convention, the boundary of any 0-simplex, $[v_0]$, is trivial, i.e. $\partial([v_0]) = 0$.

In order to analyze a complex in a particular dimension n we can form n-chains, i.e. combinations of the n-simplices in the complex. If the n-simplices are $s_1, \ldots, s_k$ then the n-chains are of the form $c = \sum_{i=1}^{n} a_i s_i$ with each a_i being either 0 or 1. For example, in a complex representing function connections a 2-chain would be a set (or sum) of triples of functionally-connected brain regions. The boundary map of a chain is then simply the sum of the boundary maps $\partial(c) = \sum_{i=1}^{n} a_i \partial(s_i)$, resulting in a lower-dimensional $(n-1)$-chain, and it turns out that the boundary map is what is called a *linear* map (which will be important later when we analyze the boundary map with common tools from the field of linear algebra, see Axler, 1997 for relevant definitions).

We can now describe n-dimensional holes in a complex, beginning with an example. Consider the left-most component of the complex in Figure 1.2, with labelled vertices as shown in Figure 1.3. Intuitively, a loop is a 1-dimensional hole that is not "filled in". Good potential candidates for loops would be 1-chains, i.e. collections of edges in the complex, with no boundary ($\partial = 0$). In Figure 1.3, four examples of such chains (labelled with colored edges) would be

$$
\begin{aligned}
c_1 &= [v_0, v_1] + [v_1, v_3] + [v_3, v_2] + [v_2, v_0] \\
c_2 &= [v_1, v_3] + [v_3, v_4] + [v_4, v_6] + [v_6, v_1] \\
c_3 &= [v_1, v_0] + [v_1, v_2] + [v_2, v_3] + [v_3, v_4] + [v_4, v_6] + [v_6, v_1] \\
c_4 &= [v_4, v_5] + [v_5, v_6] + [v_6, v_4]
\end{aligned}
$$

The first chain c_1 goes around the two filled-in triangles and should not be considered a loop as its interior is filled in, the second chain c_2 contains the middle hole and therefore should be considered a loop, the third chain c_3 winds through the filled-in triangles and then around the same hole as c_2 and therefore should be considered the same loop, and

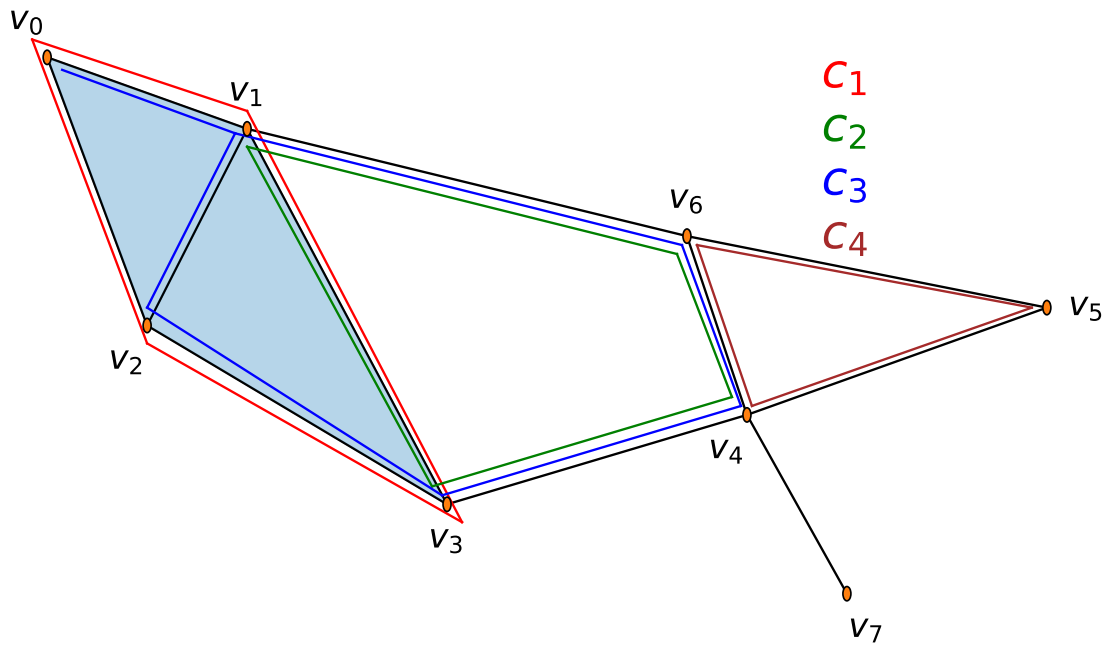

Figure 1.3: The left-most complex in Figure 1.2 with labelled vertices. The four example chains, c_1 (red), c_2 (green), c_3 (blue) and c_4 (brown) overlay the complex, following the edges of which they are each comprised.

the fourth chain c_4 contains the right hole and therefore should be considered a different loop from c_2 and c_3.

We can capture this exact intuition with boundary calculations. We see that c_1 is the boundary of the 2-chain $[v_0, v_1, v_2] + [v_1, v_3, v_2]$ (remembering that the ordering of vertices in an n-simplex does not matter):

$$
\begin{aligned}
&\partial([v_0, v_1, v_2] + [v_1, v_3, v_2]) \\
&= \partial([v_0, v_1, v_2]) + \partial([v_1, v_3, v_2]) \\
&= [v_0, v_1] + [v_1, v_2] + [v_2, v_0] + [v_1, v_3] + [v_3, v_2] + [v_2, v_1] \\
&= [v_0, v_1] + [v_1, v_3] + [v_3, v_2] + [v_2, v_0] + ([v_1, v_2] + [v_2, v_1]) \\
&= [v_0, v_1] + [v_1, v_3] + [v_3, v_2] + [v_2, v_0] + 0 \\
&= c_1
\end{aligned}
$$

On the other hand, none of the other three chains are the boundaries of some 2-chain, simply because there is no 2-simplex which contains either $[v_6, v_1]$ or $[v_5, v_6]$. Mathematically, the k-th homology group of a complex, H_k, is a quotient of two vector spaces, the kernel of the boundary map operating on k-chains $ker(\partial_k)$ (i.e. the subspace of all k-chains with trivial boundary) and the image of the boundary map operating on $(k+1)$-chains $Im(\partial_{k+1})$, i.e. $H_k = ker(\partial_k)/Im(\partial_{k+1})$. In other words, n-dimensional holes are n-chains which have 0 boundary and which are *not* the boundary of any $(n+1)$-chain.

In the quotient space H_k we remove from each k-chain with trivial boundary all its parts which are the boundary of some $(k+1)$-chain. Since $[v_1, v_0] + [v_0, v_2] + [v_2, v_0] = \partial([v_0, v_1, v_2])$ (because edges are undirected) once we remove this part of c_3 we are left with exactly c_2, and that is why they are the same loop. On the other hand, the difference between c_2 and c_4 are all of the edges except for $[v_4, v_6]$, and these edges are not a boundary, and therefore these holes are different or "independent". This intuition is formalized by the quotient space H_k – two holes are considered the same if they only differ by some boundaries of $(k+1)$-chains.

We can now summarize the "shape" of a complex by counting the number of independent holes as the dimension of the homology groups H_k, and these are called the *Betti numbers*, β_k. For instance, β_1, the first Betti number, is the number of independent loops in the complex, which for our example would be $\beta_1 = 2$. On the other hand, β_2 turns out to

be the number of 3D voids (i.e. 2-dimensional holes found inside a hollow 3D object) in a complex – since our example complex lives in 2D space, $\beta_2 = 0$. Another Betti number of interest is the 0-th Betti number, β_0, which counts the number of connected components in the complex. In our example $\beta_0 = 1$. To see why β_0 counts connected components lets look at H_0. A 0-chain is a sum (i.e. collection) of vertices. The kernel of ∂_0 is the set of 0-chains which have 0 boundary, which by definition is all chains (since each vertex has boundary 0). On the other hand any two vertices in the same connected component represent the same cycle in H_0, since they differ by the boundary of any path of edges which join them. Two vertices in different connected components cannot represent the same cycle in H_0 because they are not connected by any path of edges. Therefore, H_0 counts the number of connected components in a complex. In terms of linear algebra, β_k for any k can be calculated as dim null$(\partial_k) -$ rank(∂_{k+1}), where dim null is the dimension of the kernel and rank is the dimension of the image.

We have already discussed why complexes may be more biologically appropriate models of neurological data compared to networks, due to their ability to encode high-dimensional interactions between multiple functional units, but Betti numbers of complexes also provide several advantages over graph metrics of networks in the context of neuroimaging analyses. Most significantly, Betti numbers of a complex are not affected by the number of vertices or edges in the complex, whereas these quantities do impact the calculations of graph metrics (Rubinov and Sporns, 2010). As well, graph metrics (outside of connected components, which are the same as β_0) summarize the whole graph, whereas independent holes could capture differences in neural dynamics in distinct functional subnetworks. Also, homology disregards topologically uninformative interactions which could be desirable from a noise filtering perspective, whereas graph metrics use all the edges of a graph in their computations.

Despite these advantages, building complexes from data still falls prey to one of the most challenging problems in network neuroscience – how to decide which interactions (edges in the networks or n-simplices in the complexes) are important to include in our

model. Edges in functional connectivity networks are often filtered based on the significance of their correlations (for example using the Bonferroni correction), and this filtered network only captures network structure at the scale of the filtering threshold. However, there is no consensus on what is the best method for determining the threshold. Weighted networks on the other hand include all edges, and therefore may contain many false positive edges.

In any event, the omission of real edges or the inclusion of non-real edges has the ability to skew any graph metric or Betti number calculation. Developed independently, but of particular interest in solving this problem, a tool has been developed for calculating homology on complexes which are built from a dataset at multiple scales. This process avoids the need for (arbitrary) thresholds of (weighted) n-simplices, and can detect holes in the dataset at various scales. This tool is called *persistent homology*, and we will introduce it in the following section.

1.2.2 Capturing the shape of data with persistent homology

Persistent homology (or PH for short) is the earliest, and perhaps still most popular, tool from the field of *topological data analysis* (TDA). The original motivation for PH was topologically simplifying 3D point clouds for computer graphics tasks (Edelsbrunner et al., 2000), but it has since found applications in myriad of domains. For example, see Carlsson et al., 2007 for an application in computer vision, Yen and Cheong, 2021 for an application in economics, Krishnapriyan et al., 2021 for an application in chemistry and Haim Meirom and Bobrowski, 2022 for an application in natural language processing. Certainly this widespread adoption of PH is due to PH's ability to find non-linear structures in distinct parts of datasets, in different dimensions and across different scales (as these types of insights would be valuable for any dataset). However, another selling-point of PH is that it has but few parameters (Salch et al., 2021), each of which is easily interpretable (for instance, one parameter is the highest dimension in which to compute homology).

In the domain of visual neuroimaging PH has already been applied with widespread success, primarily in two directions – characterizing neural population codes and distinguishing between groups of functional connectome networks. The neural code direction is perhaps more validating for the use of PH in neuroimaging studies, because in certain scenarios there exists known ground truths against which we can compare estimated topologies. For instance the representational geometry of the responses of orientation-selective cells in V1 is a loop (Singh et al., 2008). On the other hand, functional connectomic analyses with PH directly solves the problem of needing to select a threshold for edge weights because PH calculates the homology of a dataset at multiple scales (i.e. edge weight thresholds). Topological features of connectomes also have the straightforward and compelling interpretation of being complex functional networks – for example a loop in a functional network could represent parallel pathways of computation which diverge and later re-converge (Sizemore et al., 2018). The next two paragraphs will provide examples of PH applications in neural codes and functional connectomes respectively, but for a general review of PH applications in fMRI across both areas (as of 2021), see Salch et al., 2021.

Persistent homology has been used to correctly identify topological features in neural population codes in both simulation studies and neural recordings. The stimulation study Ellis et al., 2019 demonstrated that a task-correlated ring of spatial activity, embedded into the responses of selected voxels in simulated event-fMRI data, could be reliably detected by PH. Without such a result, the use of PH in neuroimaging studies would be highly unmotivated – a loop is the simplest periodic structure, and one that we would expect to arise in block-design task fMRI data due to spatially-correlated signal in fMRI data and repeating tasks. Another simulation study, Kang et al., 2021, demonstrated that in a mixed population of simulated neurons the distinct topologies of the representational spaces of the different cell types could be separated. This study used a method called *persistent cohomology* (PC), which is equivalent to PH but much faster (de Silva et al., 2011a) (this method will be discussed later on). This result is important because most neuroimaging

studies collect data across diverse cell types. The studies which analyzed neural recordings on the other hand were able to be compared against known biological ground truths. In Giusti et al., 2015, Betti curves, i.e. the Betti numbers of the thresholded complexes at multiple edge-weight scales, were used to show that correlations between recordings of rat hippocampal place cells contain geometric (i.e. non-random) information. The study Singh et al., 2008 used PH to determine that the population activity (measured with micro-machined electrode arrays) representational geometry of macaque V1 cells is consistent with a 2-sphere (based on the interplay between orientation-preference and spatial-frequency maps). The approach in Chaudhuri et al., 2019 fit a low-dimensional embedding to sampled points in a neural state space, where the dimension of the embedding was determined by the Betti numbers of the dataset. This method found a ring in thalamic neuron binned spike counts data collected from moving mice, and this loop encoded head direction (a single neurologically-relevant variable in an originally high-dimensional complex data space). Taken together these studies indicate that (a) topological features can be important features of neural population codes, and (b) that PH can detect these features. A good review for the motivations for, and applications of using PH to study neural codes is Curto, 2017.

Persistent homology has also been used to distinguish between groups of functional connectomes. All of the following studies compute 0-dimensional homology of functional networks using PH, which is related to the global-scale arrangement (i.e. clustering) of vertices in the networks. The study Lee et al., 2011 found topological differences in the resting-state PET functional connectomes of children with autism, ADHD and healthy controls. In Gracia-Tabuenca et al., 2020, topological differences in resting-state connectomes were found between children with ADHD and controls, and a similar study, Gracia-Tabuenca et al., 2021, found that resting-state connectomes of children were more segregated (i.e. clustered) when undergoing puberty compared to beforehand. Studies such as these demonstrate the potential clinical utility of PH for differentiating between patient subgroups based on non-invasive and non-task-specific neuroimaging. For re-

views that introduce PH from the perspective of network neuroscience, see Sizemore et al., 2019 and Giusti et al., 2016.

Now that we have seen why PH is an appealing tool for the analysis of neuroimaging data, and that it has been applied successfully in these types of studies, let us now explain the computational process. The workflow of PH is described mathematically in Zomorodian and Carlsson, 2005 and de Silva et al., 2011b, and all the definitions in the next few paragraphs, unless otherwise stated, are taken from those sources. We will demonstrate PH pictorially with an example based on RSA in Figure 1.4, performing homology calculations up to H_1 (loops). In this example, data points are 2D stimulus representations and distances between data points are representational dissimilarities, and we will use this RSA terminology throughout our example (in general, the PH algorithm works with any data points and distances between them). In our example, PH takes as input a RDM (i.e. a distance matrix), but PH can also take as input raw representations (i.e. a point cloud dataset from which distances between data points can be calculated). The output of PH is a shape descriptor, called a *persistence diagram* (Cohen-Steiner et al., 2007), of the topological features in the representational geometry (i.e. dataset). A persistence diagram contains one 2D point for each topological feature (and thus the diagram can be visualized as a 2D scatterplot), and we computed the persistence diagram for the dataset of Figure 1.4 in Figure 1.5.

There are three main steps to computing PH:

1. At various connectivity thresholds ϵ we construct a complex whose vertices are the stimulus representations and whose n-simplices are determined by the representational dissimilarities which are at most ϵ. For instance, an edge exists between two representations if they are at most ϵ distance apart in the representational space, and a triangle exists between three representations if all of the three representational dissimilarities are at most ϵ. We can visualize this process as placing a ball of radius ϵ around each representation and connecting representations which are in eachother's balls. Each such complex is called the *Vietoris-Rips* (VR) complex at the scale ϵ, de-

noted VR_ϵ in this thesis, and can be viewed as an estimate of the topological (i.e. dataset) structure at scale ϵ.

2. We grow the parameter ϵ from 0, computing a sequence of VR complexes at multiple ϵ values in a process called a *filtration*. This is how we can study the representational geometry at multiple connectivity scales simultaneously.

3. Each VR complex contains topological features which can be identified by homology calculations, for instance clusters of connected representations are elements of H_0, loops of representations are elements of H_1, etc. We can therefore track the existence of these features over consecutive ϵ values to identify topological features of the representational space which persist (i.e. exist) over multiple connectivity thresholds. The connectivity scale ϵ at which a particular topological feature, for instance a loop of representations, comes into existence is called the feature's *birth* radius. Since ϵ values are grown in the filtration, later VR complexes will always contain all of the n-simplices in earlier complexes, so if ϵ is grown large enough then any hole will eventually get filled in by the added (long-range) connections. The ϵ value at which a topological feature gets filled in (i.e. "dies") is called its *death* radius. Each topological feature is thus assigned a 2D point, which is $(\text{birth radius}, \text{death radius})$, and the collection of all these points is the persistence diagram.

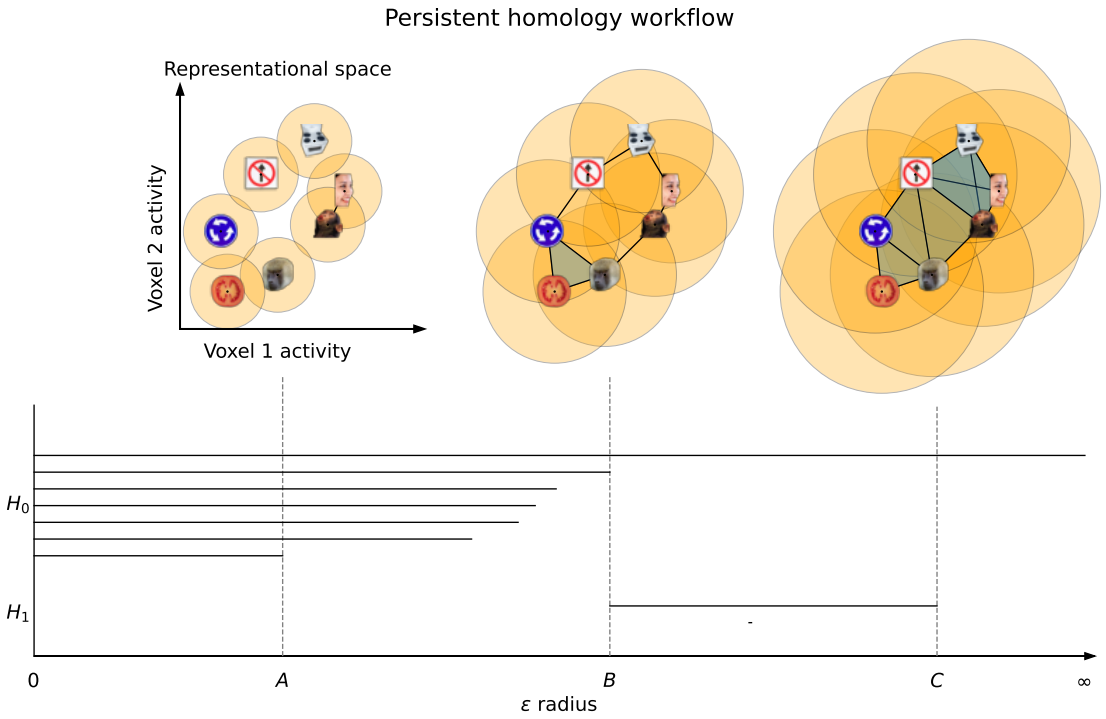

Figure 1.4: The workflow of persistent homology (copied from Figure 3.2). In this example we have a dataset of stimuli (these are a subset of the stimuli used in the famous visual RSA study Kriegeskorte, 2009) in a simulated fMRI 2D representational space. Each stimuli is plotted at its coordinates in the space, with a small dot at its coordinates. We then grow an ϵ parameter from 0 and visualize the constructed complex at three ϵ values, A, B and C. For example, the complex at radius A has balls of radius A plotted around each stimulus, and stimuli are connected when their two balls contain both stimulus points (and triangles are formed between triples of connected stimuli). We construct lines which encode the lifespan of all the topological features of the sampled points, for which A, B and C represent certain ϵ values at which certain features either are born or die. At radius A there are six "living" H_0 components – since the human and monkey faces are connected they form a single component, and the other five stimuli are their own components. At radius B the components all merge into a single cluster, and a loop is born. At radius D the loop dies, leaving only one cluster which lives "forever" (i.e. we can increase the linkage radius to any larger value and there will remain one cluster). The persistence diagram for this example dataset can be seen in Figure 1.5. The code for this plot was inspired by https://github.com/iaciac/py-draw-complex/blob/master/Draw%202d%20simplicial%20complex.ipynb.

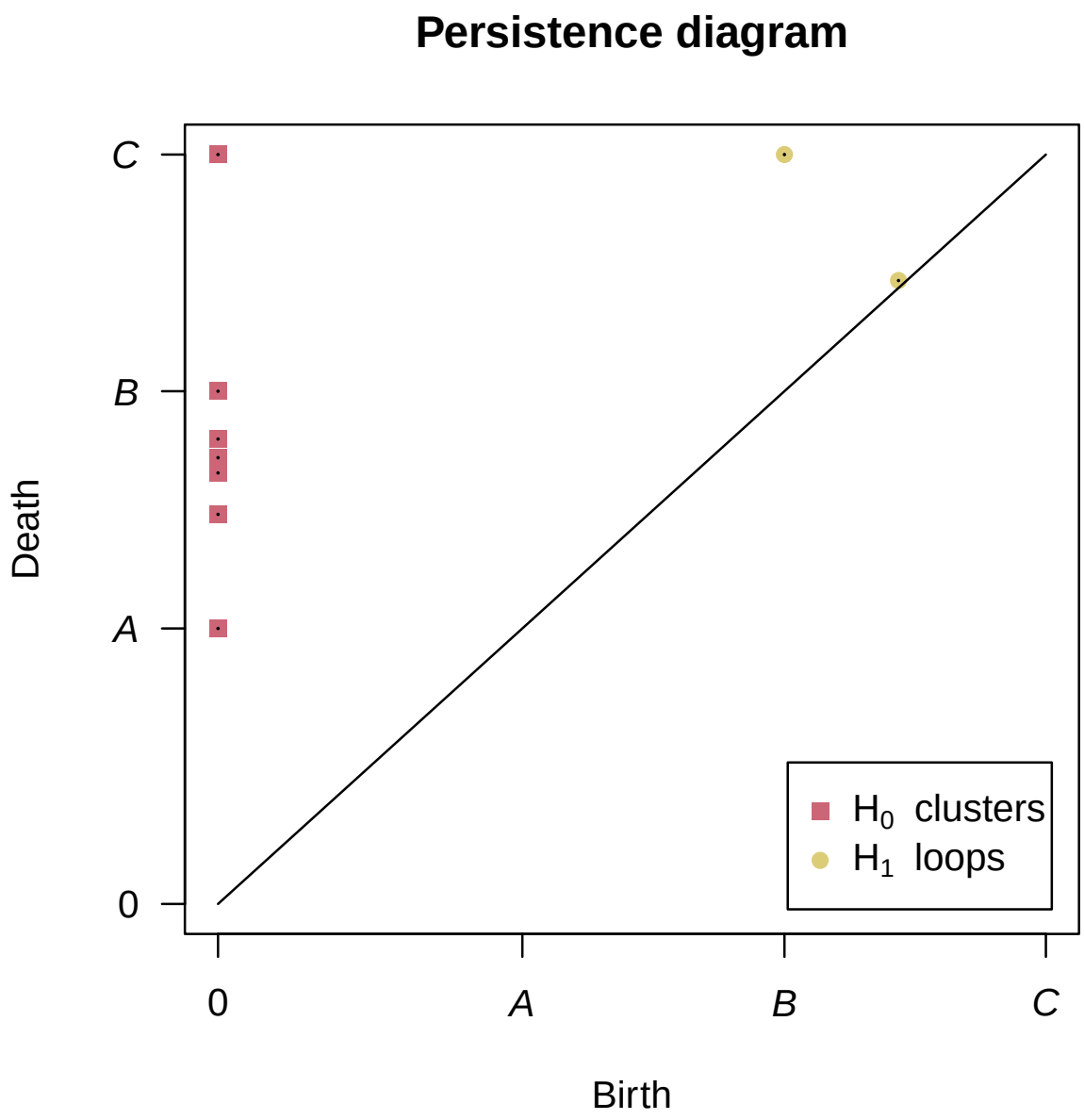

Figure 1.5: The computed persistence diagram for the dataset in Figure 1.4, taken from Figure 3.3. The x-values of this plot are the birth radii of topological features and the y-values of this plot are the death radii. Reddish points denote H_0 components and the gold points represent H_1 loops in the dataset. The main loop in the dataset, as seen in Figure 1.4, is the gold point high above the diagonal, and a very short-lived loop (represented by the tiny line segment below the long lasting loop's line segment in Figure 1.4) is the other gold point.

We described in the previous section how the homology of a single complex can be calculated using the boundary map and tools from linear algebra. A similar approach is also used to compute persistent homology: let $S = \{s_1, \ldots, s_k\}$ be all of the simplices that arise in the filtration process, ordered by their birth radius. We compute the "boundary matrix", D, whose rows and columns correspond to the elements of S, and $D[i, j] = 1$ if s_i is in the boundary of s_j, $\partial(s_j)$, and $D[i, j] = 0$ otherwise. This matrix encodes both the boundary calculations and the filtration process (since S is ordered according to the filtration). We can then "reduce" this matrix using certain linear column operations to obtain a basis for the independent dimensional holes of the filtration for each dimension n. Basis elements for n-dimensional holes are defined by a pair of n-simplices (s_i, s_j), where s_i completes the basis element (for example the edge that completes a loop) and s_j destroys the basis element (for example the edge that fills in the interior of a loop), and the birth and death radii of this feature are the ϵ radii where s_i and s_j appear in the filtration respectively. A faster algorithm for computing these "persistence pairs" (i.e. the persistence diagram) reduces D using linear column (as opposed to row) operations, and this process is the persistent cohomology algorithm. The details of these algorithms can be found in Zomorodian and Carlsson, 2005 and de Silva et al., 2011b.

The most common way to interpret persistence diagrams is to calculate what is called the *persistence* of each topological feature, which is its death radius subtract its birth radius. In the persistence diagram, persistence values give the vertical distance from each point to the diagonal line where birth and death are equal to each other (points close to this line can be thought of as topological features which are noise as they appear and then disappear quickly). Therefore, features with larger persistence values likely represent "real" or "significant" features of the representational space, and those with smaller persistence values likely represent topological noise. By filtering topological features by a persistence threshold, we can determine the number of significant features in each homological dimension (i.e. the number of clusters, loops, etc.) and these can be thought of as the "Betti numbers" of the data space (like a representational space).

On the other hand, the birth and death radii can also be interpreted in other ways. The first view, based on how persistent homology is calculated, is that the birth and death radius of a topological feature are the scales at which the feature first appears and first ceases to exist, respectively, and that these values provide information about the "size" of the feature in the dataset. Another interpretation – one that I have not seen elsewhere – is that birth radius is related to the sampling rate of the topological feature and that death radius is related to the size of the feature. As more points get sampled from a feature, the points will become closer together and therefore the birth radius will decrease (and vice versa). On the other hand, the feature will exist until we connect points just far enough *across* the feature, and this term *across* is really just dependent on the size of the feature.

Persistent homology has several desirable qualities which make its application in neuroimaging studies very appealing. The information in a neural code is not altered by the relabelling of functional units (for instance deciding that voxel 1 is actually voxel 2 and vice versa) (Laakso, 2000), and this is also a property of PH because PH only depends on the distances between points (the order of which does not affect the output persistence diagram). On the other hand, the information of a neural code is not altered by the scale of neural activity (if we multiplied all voxel activity by 2 the neural code would remain the same) (Laakso, 2000), and this is also a property of PH because scaled distances would only result in a scaled persistence diagram (containing the same number and type of features as before). Finally, the information of a neural code is not altered by rotating and reflecting (in its representational space) (Laakso, 2000), operations which would preserve all representational distances. The fact that PH depends only on the ordering of representational dissimilarity values (as this ordering fully determines the order in which simplices appear in the filtration process) as opposed to the values themselves reflects the previously mentioned property of the theory of pattern activations, that only the *relative* position between representations in state space are important Laakso, 2000. In Kriegeskorte and Kievit, 2013, a review of using representational geometry (i.e. RSA) to study neural population codes, it was asked, "which mathematical concepts from topol-

ogy and geometry might be useful for understanding neuronal population codes?" Persistent homology has all of the building blocks required to be a powerful tool for analyzing representational geometries of neural codes.

Persistence diagrams are one of the most common output formats of PH, but other options do exist. A popular alternative is called the *persistence barcode*, which represents each topological feature as the interval $(\text{birth radius}, \text{death radius})$ (this can be seen as an intermediate step to calculating a persistence diagram in Figure 1.4, and is equivalent in practice to persistence diagrams). However, one difficulty with these two data formats, persistence diagrams and barcodes, is that they are a unique form of unstructured data – a collection of some number of pairs of values. The persistence diagrams (or barcodes) of two representational spaces need not contain the same number of topological features, even if the spaces contain the same number of stimulus representations. This makes statistics (like means) and distributions of persistence diagrams complicated to estimate (Turner, 2013; Turner et al., 2014).

In order to avoid these kinds of issues, a number of fixed-length vectorized summaries of persistence diagrams have been introduced. Collection of vectors have well-defined, and easy to compute, means and distributions. The most simple vector summary is called *Betti curves*, that is, functions $\beta_k(\epsilon)$ which compute the Betti number β_k in the rips complex VR_ϵ. These functions have one value (the Betti number) at each ϵ value in the filtration, and hence multiple Betti curves can be easily compared using typical vector operations. More complex vectorizations of persistence diagrams include *persistence landscapes* (Bubenik, 2015; Bubenik and Dlotko, 2017), *persistence silhouettes* (Chazal et al., 2014) and *persistence surfaces* (Adams et al., 2017).

Persistence landscapes are functions $\lambda_k(\epsilon)$ which calculate the maximum deviation v from the connectivity radius ϵ for which between the connectivity thresholds $\epsilon - v$ and $\epsilon + v$ there are at least k topological features in the persistence diagram. Therefore, each λ_k function can be represented as a vector, whose length is the number of ϵ values over which we compute our PH filtration and the values of the vector are the function values

$\lambda_k(\epsilon)$ for each ϵ value. Like Betti curves, persistence landscape functions can be easily compared (for instance, calculating a "mean" landscape function is perfectly reasonable) so long as the same filtrations (or even just the number of connectivity threshold values) are used to compute PH across multiple datasets.

Persistence silhouettes were built off persistence landscapes – we start by defining, for each point $p = (b, d)$ in a persistence diagram D its triangle function $\Lambda_p(\epsilon)$ which is 0 if ϵ is not between the birth and death values b and d, and otherwise is the distance from ϵ to the closer of b and d (this forms a graph which looks like a triangle over all ϵ values). We then construct persistence silhouette functions as weighted sums of the triangle functions (with weight w_p for point p): $\phi(\epsilon) = \frac{\sum_{p \in D} w_p \Lambda_p(\epsilon)}{\sum_{p \in D} w_p}$. These functions are as easily compared as persistence landscapes, and are useful because they can disregard (via smaller weights) topological features which are less persistent.

Persistence surfaces take a very different approach – a distribution is created from a persistence diagram by adding a (weighted) 2D Gaussian point-mass of variance σ^2 centered at each 2D point in the diagram. The distribution can be defined on a grid of 2D points, with PDF value of the distribution at a 2D point proportional to the sum of the PDF values of each Gaussian point mass at that location. This approach also creates a fixed-length vector (of distribution values over the grid) which is easily compared across persistence diagrams, but has the additional access to tools from probability theory for analyzing persistence surfaces (one of these tools we will see later).

Vectorized summaries of persistence diagrams, such as Betti curves, persistence landscapes, silhouettes and surfaces, are useful representations for statistics and machine learning due to their structured nature, fast calculation and efficient comparisons (Ali et al., 2023; Hensel et al., 2021). However, as summaries of persistence diagrams these vectors cannot contain more information than the persistence diagrams they came from, and certainly could result in the loss of topological information. Some of these methods may also be more challenging to interpret, like the multi-layered definition of persistence landscapes. Finally, each method requires additional computational steps which could

complicate analysis pipelines. For these reasons we will only concern ourselves with analyzing persistence diagrams (computed from representational geometries) in this thesis.

While persistence diagrams do not easily lend themselves to statistics and machine learning analyses, that is not to say that this kind of machinery has not been developed. In the following section we will describe several published methods for analyzing persistence diagrams with machine learning and statistical inference, along with the machinery we need to carry out these procedures.

1.2.3 Interpreting and analyzing persistence diagrams

Persistent homology has the ability to capture shape structures in representational spaces which other techniques miss – even non-linear techniques (Chung and Abbott, 2021). These features capture local to global scale clustering of any shape, loops of 1-dimensional periodic structure, voids of 2-dimensional periodic structure, etc. Unfortunately there are two main challenges in PH analyses. Firstly, how can we interpret the computed features? Knowing that a loop exists somewhere in our dataset is interesting, but without knowledge of its location in data space we cannot convert this loop into a meaningful feature of data points. And even more sinister is that this loop may not represent a "significant" structure in our dataset, rather representing topological noise. Secondly, since persistence diagrams are unstructured data, how can we use them in downstream analyses? In RSA multiple RDMs can simultaneously be projected, using multidimensional scaling, into a 2D space which captures the second-order distances between the RDMs, but it is not obvious what would be an appropriate second-order isomorphism of persistence diagrams to carry out an analogous analysis. In fact, all statistics and machine learning analyses of RDMs in RSA are based on second-order isomorphisms, i.e. quantifying difference and similarity, between RDMs.

Thankfully there are a number of tools that have been developed to solve exactly these types of problems for persistence diagrams, which we will overview here and describe more rigorously in Chapter 2. For second-order isomorphisms, the *bottleneck* and *wasser-*

stein distance functions (Kerber et al., 2017) (or "metrics" in the math literature) are popular methods for quantifying differences between persistence diagrams, and the *persistence Fisher kernel* (PF) (Le and Yamada, 2018) is a useful method for quantifying similarity between persistence diagrams. These distance functions have opened the door to some statistical inference methods, including bootstrap filtering the topological features in a persistence diagram for significance based on their persistence values (Fasy et al., 2014). Distance functions have also been applied in a hypothesis testing procedure for finding differences between groups of persistence diagrams (Robinson and Turner, 2017) akin to a non-parametric ANOVA test. One paper, Abdallah et al., 2023, even refined the group-difference procedure for fMRI data (i.e. reduced variance and increased statistical power) by accounting for temporal dependencies that exist within task blocks. On the other hand, special similarity functions called *kernels* open the door for carrying out machine learning on unstructured data (Murphy, 2012). The PF kernel has been used to fit support vector machine (SVM) (Murphy, 2012) models to successfully distinguish between different datasets based on their persistence diagrams (for example distinguishing images of different classes of shapes, with multiple examples from each shape category), essentially by correlating kernel values between pairs of diagrams with the similarity of their labels (in the shape dataset example the labels would be the shape category of each diagram). There are a number of machine learning algorithms which are based on distance functions (e.g. MDS) or kernel functions (e.g. kernel k-means (Dhillon et al., 2004) for clustering and kernel principal components analysis (Scholkopf et al., 1998) for dimension reduction). Based on these applications, we will propose the bottleneck, wasserstein and persistence Fisher functions as useful second-order isomorphisms of persistence diagrams.

The persistence Fisher kernel is not the only, nor the first kernel function, for persistence diagrams that has been developed; however, we will focus on the PF kernel because it was found to outperform other kernels in a number of SVM decoding tasks. We will define and describe the persistence Fisher kernel in Chapter 2, but for the sake of comparison we will briefly introduce three other kernels for persistence diagrams. The original

kernel for persistence diagrams is called the *persistence state space kernel* (PSS kernel, Reininghaus et al., 2015), and captures the idea that two diagrams should be similar (i.e. kernel value closer to 1) if Gaussian point masses of variance σ^2 centered at each point of the first diagram find the points in the second diagram highly likely and that the diagonal projection of the points in the second diagram (i.e. the set of points on the diagonal line $y = x$, where birth and death are equal, that are closest to each point in the second diagram) very unlikely. Then, the *persistence-weighted Gaussian kernel* (PWGK, Kusano et al., 2018) allows the weighted contribution of Gaussian probabilities based on the persistence values of the 2D points being compared, thereby filtering out some topological noise. *Variably-scaled persistence kernels* (VS kernel, De Marchi et al., 2022) rescale the importance of points in each diagram before the kernel calculation (rather than multiplying a kernel calculation by weights in the persistence-weighted Gaussian kernel). These kernels are provably stable, i.e. small perturbations to the points in two input persistence diagrams result in similar kernel values, but the *sliced wasserstein kernel* (SW kernel, Carrière et al., 2017) is discriminative in that differences in kernel values correlate with difference in wasserstein distance values between diagrams, making similarity and differences essentially opposites using this function. All of these methods have computational complexity (at least) $O(n^2)$, where n is the larger number of topological features in the two input persistence diagrams, which is a significant computational burden when the diagrams contain large numbers of features. However, all three methods also have efficient approximation methods which increases their practicality for application. We will see later that the PF kernel also has a fast approximation method.

While the bootstrap method for filtering persistence diagrams can help identify significant topological features in a dataset, we need additional tools to determine what the features represent in data space – i.e. there may be a significant loop *somewhere*, but *where* is it and *what* does it mean? *Representative cycles* and *representative co-cycles*, byproducts of the original PH and PC algorithms, can provide a subset of the data points which exist on a particular topological feature, thereby providing hints as to the location of the feature.

We can then hone in on and visualize the feature using *Vietoris-Rips graphs* (VR-graphs, Zomorodian, 2010), which is the graph formed by persistent homology at a particular connectivity radius ϵ (i.e. the 1-skeleton of VR_ϵ). For example, the VR graph of a dataset at the scale of a loop birth radius will contain the loop (with the minimal possible number of edges, easing visualization), and by highlighting the representative data points we can identify where the loop exists in the data. The ordering of data points around the loop can then be used to help determine what the feature of the data the loop represents. For instance, in the representational geometry of a visual fMRI study a loop may represent stimulus orientation. Using all three tools – the bootstrap, representative cycles and VR graphs – we can segment representational geometries, i.e. RDMs, into distinct non-linear shape features of various dimensions, and we will see in Chapters 2 and 3 examples of how these topological features can be related to stimulus features in (RSA) studies, thereby linking brain to stimulus via topology.

1.2.4 Applied topological analyses in R and Python

In Chapter 2 we will introduce our new R software toolbox, `TDApplied`, for applied topological analyses of data via statistical inference and machine learning of persistence diagrams. `TDApplied` provides the functionality to carry out the procedures described in the previous section as well as others. However, there are a number of other R and Python packages for topological data analysis.

In R the main packages for topological data analysis are `TDA` (Fasy et al., 2021) and `TDAstats` (Wadhwa et al., 2018), although `TDA` is currently unavailable on CRAN (the main repository for publicly available R packages). Both packages can compute persistence diagrams by wrapping `C++` libraries – `TDA` wraps `dionysus` (Morozov, 2017), `PHAT` (Bauer et al., 2013) and `GUDHI` (Lacombe et al., 2019), and `TDAstats` wraps `ripser` (Bauer, 2015), and another package which is currently unavailable on CRAN called `rgudhi` (Stamm, 2023) also wraps `GUDHI`. `TDA` is able to compute distances between persistence diagrams and can therefore run the bootstrap procedure from the previous section.

`TDAstats` has a function to find group differences of persistence diagrams (as described in the previous section); however, it uses a non-published distance function which makes interpretation of the inferences from the procedure challenging. The R packages `TDAkit` (You and Yu, 2021) and `TDAvec` (Islambekov and Luchinsky, 2022) are designed to extract and analyze feature vectors (for instance persistence landscapes) computed from persistence diagrams.

In Python there two libraries which encompass a number of software packages for topological data analysis – `scikit-tda` (Saul and Tralie, 2019) computes persistence diagrams and distances/kernels between them, and `giotto-tda` (Tauzin et al., 2020) which can compute persistence diagrams and interfaces with the popular package for machine learning `scikit-learn` (Pedregosa et al., 2011) for using vectorized representations of persistence diagrams in machine learning pipelines.

Despite the wide-scale applicability of these software packages none provide the functionality to perform machine learning and inference directly with persistence diagrams, i.e. without first vectorizing the diagrams. This is a significant gap in the topological data analysis software landscape, since persistence diagrams are one of the most popular and interpretable outputs of PH, and inference and machine learning are two of the most popular data analysis frameworks. Out of the analysis procedures listed in the previous section only the `TDA` package can perform the topological bootstrap, only `TDA` and `scikit-tda` can calculate representative (co)cycles and only `giotto-tda` can carry out machine learning on vectorized persistence diagrams; none of the other procedures are carried out by any of the packages. `TDApplied`, as described in Chapter 2, fills this gap in order to provide researchers and data professionals with the ability to carry out efficient applied topological analyses of persistence diagrams, with a number of inference and machine learning procedures as well as tools for interpreting persistence diagrams.

1.2.5 Other topological tools

Persistent homology is the first, and perhaps most popular, tool in the toolbox of TDA, but another method called *mapper* (Singh et al., 2007) has also been successfully applied in a number of neuroimaging studies. The mapper algorithm reduces a high-dimensional dataset to a graph, where each node of the graph represents a cluster of similar data points, and edges between two nodes represent shared data points between the two clusters (some clusters will overlap by construction). Graphs are low-dimensional objects which are well-studied and well-used in neuroimaging research, making mapper an appealing tool for computing topological summaries of neurological data. Mapper is therefore a more easily interpreted topological summary compared to persistent homology, and due to the clustering procedures used in mapper, its graphs represent more compressed (and flexible) representations of data than VR graphs.

Mapper has been the primary tool of analysis in a number of high-profile publications, two of which are Saggar et al., 2018 and Saggar et al., 2022. In Saggar et al., 2018, mapper graphs were computed from task fMRI data in subjects performing multiple tasks (visuospatial search, arithmetic and memory) within the same scan, where the data points were spatial activity patterns at individual time points either during one of the tasks or during a period of rest, and the graph structure was found to be correlated with behavior and to encode task structure. The modularity of the computed mapper graphs, i.e. how separated the task representations were, was correlated with task performance – the more specialized each task computation was in the brain, the higher the performance. As well, it was found that specialized task representations were generally in the ”core” of the graphs whereas non-specialized rest representations were generally in the ”periphery”. On the other hand, specialized task representations tended to share node memberships with many other representations whereas rest representations did not, and changepoint detection algorithms was used to quickly (within a few time frames) detect transitions between tasks. In Saggar et al., 2022 it was found that resting-state brain transitions are continuous (as opposed to discrete) and structured (i.e. non-random) by analyzing

mapper graphs of resting-state fMRI data. By computing the contribution of various *resting-state networks* (RSNs), i.e. activity correlations which exist in the absence of task (Seitzman et al., 2019; Yeo et al., 2011), to each mapper graph node it was found that the distribution of RSN contributions varied smoothly across the mapper graphs, with more central "hub" nodes exhibiting more uniform distributions and being the most popular (i.e. likely) intermediate neural states along a smooth neural transition. A software called `dynamical neuroimaging spatiotemporal representations` (`DyNeuSR`) (Geniesse et al., 2019), specifically designed for analyzing and visualizing neuroimaging data with mapper, has also been published to facilitate future research.

Despite the unique ability of mapper to visualizing and analyze neuroimaging data, and therefore answer research questions that other techniques cannot, mapper does have some drawbacks compared to persistence diagrams and VR graphs. For example, mapper has number of parameters – a distance function between data points must be chosen, a clustering algorithm (with all of its parameters) must be chosen, a function which assigns a value to each data point must be chosen and two other numeric parameters must be selected. On the other hand, persistent homology only requires the distance function between data points, and the VR graph only requires the distance function and an ϵ value. This means that selecting parameters for constructing meaningful mapper graphs of neurological data may be more challenging and unstable compared to persistence diagrams and VR graphs. Clustering data points in mapper may also obscure real signal in the data by making assumptions on cluster shape or distribution. Moreover, graphs are 1-dimensional complexes, and therefore mapper cannot capture high-dimensional structure (like voids) in datasets, nor is it tailored to identify loops, and, as we discussed earlier, periodic structures are incredibly important in neurological data. It is for reasons like these that we did not use mapper graphs as the "RDM-replacement" data structure in our RTA framework.

Another class of algorithms which are sometimes viewed as topological is non-linear dimension reduction, or "manifold learning". Methods in this class generally compute

either a k-nearest neighbor graph of a dataset (i.e. a graph in which we connect each data point to its k nearest other data points) or an ϵ-neighborhood graph (i.e. a graph in which we connect each data point to all other points within distance ϵ) and then obtain a low-dimensional (non-linear) embedding of the dataset which respects proximity between data points (nodes) in the graph. Examples of these methods include Isomap (Tenenbaum et al., 2000) and Laplacian eigenmaps (Reuter et al., 2006), and there have been a number of applications of manifold learning in neuroimaging applications. In Atasoy et al., 2016, the Laplacian eignemaps of fMRI surface data were coined "surface connectome harmonics" and were found to exhibit high mutual information with a number of resting state networks and to capture expected inhibition-excitation interactions in the brain. In Gerber et al., 2009, Isomap was used to project a dataset of T1-weighted MRI images, with a tailored distance metric between data points, into low dimensions. In I.V et al., 2015, an increase in fMRI task decoding was found when preprocessing fMRI data with techniques like Iosmap and Laplacian eigenmaps.

Despite the utility of manifold learning techniques, they too were not appropriate as a topological summary of representational datasets. This is because no manifold learning technique is tailored to capture specific non-linear periodic structures in the data, unlike PH which can identify loops, voids, etc. Further, PC is able to perform non-linear dimension reduction by locating data points on the various holes that exist in the dataset, and these holes are often undetected by manifold learning techniques (de Silva et al., 2011a).

1.3 Unresolved questions

Representational similarity analysis is unable to capture real topological differences of representational geometries because topological structures can be comprised of any number of representational dissimilarities. A topologically-conscious RSA framework, which replaces RDMs with persistence diagrams and replaces Spearman correlation (distance) with distance and kernel functions of diagrams, therefore could answer certain questions

about neural function that RSA could not. However, in order for such a framework to be adopted, we must demonstrate that the new framework can carry out the same kinds of analyses as RSA does with RDMs (such as inference and machine learning) and that the new framework can identify representational differences that RSA misses. In Chapter 2 I present my software package `TDApplied` for applied analyses with, and interpretations of, persistence diagrams (such as those computed in representational spaces), showing that the new framework can carry out the same types of analyses as RSA. In Chapter 3 I formally introduce the RTA framework and provide two studies which demonstrate how RTA can identify topological features of representational geometries which RSA cannot.

In summary, `TDApplied` and the two studies demonstrate the practicality and unique value of the RTA framework as a companion tool to RSA. In Chapter 4 I will discuss the avenues of research that are now possible in the RTA framework.

Chapter 2

Machine learning and inference for topological data analysis with `TDApplied`

2.1 Preamble

The motivation for this software originally was my desire to use the group differences inference test of Robinson and Turner, 2017 from the `TDAstats` (Wadhwa et al., 2018) package for analyzing persistence diagrams computed from RDMs, and realizing that the distance function implemented in `TDAstats` was non-standard. Therefore I set out to write the group difference function myself from scratch, later adding many more functions in order to carry out particular RTA analyses I was interested in, and found development was easiest in the framework of an R package. The most popular repository for publicly available R software packages is "the comprehensive R archive network", or CRAN for short (https://cran.r-project.org). In order to facilitate the use of `TDApplied` in research applications I have released several versions of the software on CRAN, in addition to the development version always available on GitHub (https://github.com/shaelebrown/TDApplied). Essential to the usability of `TDApplied` I have included significant amounts

of examples, tests and documentation to guide users in their own applied topological analyses. The major documentation of R packages are called vignettes, which are standalone documents providing examples or explanations about particular topics related to a package, and are viewable from the package's page on CRAN (for instance, `TDAstats` has a vignette called "Introduction to persistent homology with `TDAstats`" which is viewable at https://cran.r-project.org/web/packages/TDAstats/index.html). The package vignettes are written for the audience of general R-users (i.e. with a software/data science background).

This chapter is organized as follows. First we include our paper submitted to the Journal of Open Source Software (JOSS), which is meant to be a high-level description of the software's purpose with a statement of need. The remaining sections are the various `TDApplied` vignettes, addressing (1) package functionality and examples on simulated data (called "`TDApplied` theory and practice"), (2) an example RTA-like analysis on the famous neuroimaging dataset called the *Human Connectome Project* (Glasser et al., 2016) ("Human Connectome Project analysis"), (3) computational efficiency comparisons against other r and Python packages for topological data analysis ("Benchmarking and speed"), (4) a workflow for carrying out topological analyses with inference and machine learning methods which are not in the package ("Personalized analyses with `TDApplied`") and (5) explaining differences in distance calculations across R packages ("Comparing distance calculations"). These vignettes demonstrate that `TDApplied` (1) works as expected on simulated data, (2) can carry out meaningful applied analyses of neurological data (however, note that this vignette is meant to be an example usage of `TDApplied` and not a rigorous demonstration of new neurological findings), (3) is a highly practical tool for applied topological analyses, (4) can be integrated into custom analysis pipelines and (5) has correct distance calculations (which are the basis of all applied topological analyses). The paper and vignettes therefore provide evidence that `TDApplied` is an effective tool for RTA analyses of persistence diagrams, and also for applied topological analyses in other domains.

In order to make sure that this thesis has a consistent formatting style, we have modified the contents of this Chapter in two ways (compared to the version submitted to JOSS). By convention figures in vignettes are not labelled because they immediately follow the code chunk which generated them, but figure numbers and captions have been added to this chapter for consistency. Since the domain of this thesis is neuroscience we will follow the conventional author-year citation and bibliography format for neuroscience publications in this chapter.

Submitted to JOSS, 2024.

`TDApplied`: An R package for machine learning and inference with persistence diagrams

Shael Brown[1] and Reza Farivar[2]

[1]Department of Quantitative Life Sciences, McGill University, Montreal Canada.

[2]McGill Vision Research, Department of Opthamology, McGill University, Montreal Canada.

2.2 Summary

Topological data analysis is a collection of tools, based on the mathematical fields of topology and geometry, for finding structure in whole datasets. Its main tool, persistent homology (Edelsbrunner et al., 2000; Zomorodian and Carlsson, 2005), computes a shape descriptor of a dataset called a persistence diagram which encodes information about holes that exist in the dataset (example applications span a variety of areas, see for example Gracia-Tabuenca et al., 2020; Haim Meirom and Bobrowski, 2022; Krishnapriyan et al., 2021). These types of features cannot be identified by other methods, making persistence diagrams a unique and valuable data science object for studying and comparing datasets. The two most popular data science tools for analyzing multiple objects are machine learning and inference, but to date there has been no open source implementation of published methods for machine learning and inference of persistence diagrams.

2.3 Statement of need

`TDApplied` is the first R package for machine learning and inference of persistence diagrams, building on the main R packages for the calculation of persistence diagrams `TDA` (Fasy et al., 2021) and `TDAstats` (Wadhwa et al., 2018, 2019) and publications of applied analysis methods for persistence diagrams (Le and Yamada, 2018; Robinson and Turner, 2017). `TDApplied` is intended to be used by academic researchers and industry professionals wanting to integrate persistence diagrams into their analysis workflows. An example `TDApplied` workflow, in which the topological differences between three datasets are visualized in 2D using multidimensional scaling (MDS) (Cox and Cox, 2008), is visualized in figure 2.1:

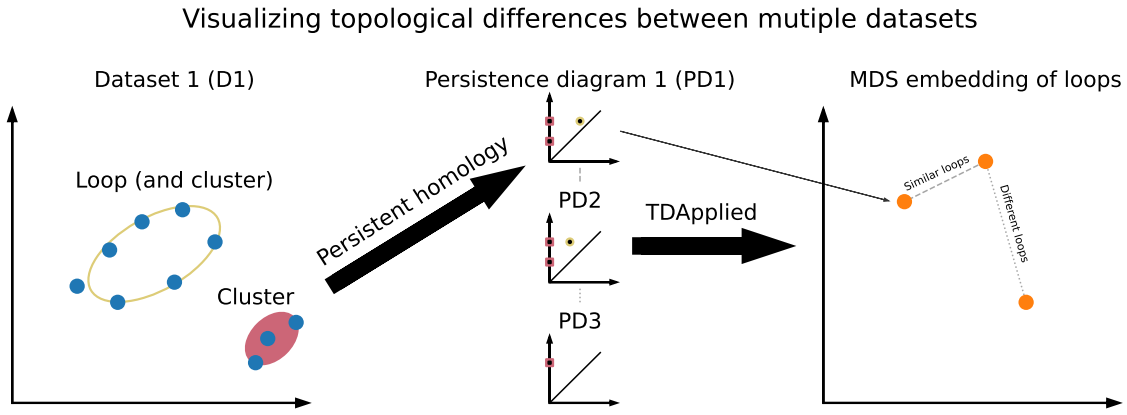

Figure 2.1: An example `TDApplied` workflow. A dataset (D1, left) contains one loop (yellow) and two clusters (the loop forms one cluster and the three points on the bottom are another cluster, and clusters are denoted by the color red). These topological features are captured with persistent homology in a persistence diagram PD1 (middle top), and two other data sets, D2 and D3 (not shown), have their persistence diagrams, PD2 and PD3, computed (middle center and middle bottom). PD1 and PD2 are not very topologically different in terms of their loops, with both containing a loop with similar birth and death values, and this is represented by a dashed-line relationship. On the other hand, PD2 and PD3 are topologically different in terms of their loops because PD3 does not contain a loop, and this is represented by a dotted-line relationship. `TDApplied` can quantify these topological differences and use MDS to project the persistence diagrams into three points in a 2D embedding space (right) where interpoint distances reflect the topological differences between the persistence diagrams.

The `TDApplied` package is built on three main pillars:

1. User-friendly – internal preprocessing of persistence diagrams that would normally be left to R users to figure out ad hoc, and functions designed to easily flow from input diagrams to output metrics.

2. Efficient – parallelization, C code, computational tricks and storage of reusable and cumbersome calculations significantly increases the feasibility of topological analyses (compared to existing R packages).

3. Flexible – ability to interface with other data science packages to create personalized analyses.

`TDApplied` has already been featured in a conference workshop (https://github.com/WoComtoQC/wocomtoqc.github.io/blob/main/abstract.md) and a conference tutorial (https://www.ihcisociety.org/program/tutorial-lecture), utilized in a journal publication (Singh et al., 2023) and downloaded over 4400 times. Therefore, we propose `TDApplied` as a user-friendly, efficient and flexible R package for the analysis of multiple datasets using machine learning and inference via topological data analysis.

2.4 Project management

Installation and availability: `TDApplied` can be installed directly from CRAN using the command `install.packages("TDApplied")`, or from GitHub using the `devtools` package (Wickham et al., 2021). `TDApplied` is distributed under the GPL-3 license.

Code quality: Code has been tested using the `testthat` package (Wickham, 2011), with 91.45% coverage of R code when not skipping tests involving Python code (or 88.44% coverage when skipping the Python tests).

Documentation: `TDApplied` contains five main vignettes:

1. "`TDApplied` theory and practice" provides example function usage on simulated data as well as mathematical background and intuition,

2. "Human Connectome Project analysis" demonstrates an applied example analysis of neurological data,

3. "Benchmarking and speedups" outlines the package's optimization strategies and highlights performance gains compared to other packages,

4. "Personalized analyses with `TDApplied`" demonstrates how to interface `TDApplied` with other data science packages, and

5. ”Comparing distance calculations” accounts for differences in computed distance values between persistence diagrams across comparable packages.

Acknowledgements: We acknowledge funding from the CIHR 2016 grant for cortical mechanisms of 3-D scene and object recognition in the primate brain.

2.5 References

Cox, M. A. A., & Cox, T. F. (2008). Multidimensional scaling. In *Handbook of data visualization* (pp. 315–347). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-33037-0_14

Edelsbrunner, H., Letscher, D., & Zomorodian, A. (2000). Topological persistence and simplification. *Discrete & Computational Geometry*, *28*, 511–533. https://doi.org/10.1007/s00454-002-2885-2

Fasy, B., Kim, J., Lecci, F., Maria, C., Millman, D. L., & Rouvreau., V. (2021). *Tda: Statistical tools for topological data analysis* [R package version 1.7.7]. https://CRAN.R-project.org/package=TDA

Glasser, M. F., Smith, S. M., Marcus, D. S., Andersson, J. L. R., Auerbach, E. J., Behrens, T. E. J., Coalson, T. S., Harms, M. P., Jenkinson, M., Moeller, S., Robinson, E. C., Sotiropoulos, S. N., Xu, J., Yacoub, E., Ugurbil, K., & Van Essen, D. C. (2016). The human connectome project's neuroimaging approach. *Nature Neuroscience*, *19*(9), 1175–1187.

Gracia-Tabuenca, Z., Diaz-Patino, J. C., Arelio, I., & Alcauter, S. (2020). Topological data analysis reveals robust alterations in the whole-brain and frontal lobe functional connectomes in
attention-deficit/hyperactivity disorder. *eneuro*. https://doi.org/10.1523/eneuro.0543-19.2020

Haim Meirom, S., & Bobrowski, O. (2022). Unsupervised geometric and topological approaches for cross-lingual sentence representation and comparison. *Proceedings of*

the 7th Workshop on Representation Learning for NLP, 173–183. https://doi.org/10.18653/v1/2022.repl4nlp-1.18

Krishnapriyan, A. S., Montoya, J., Haranczyk, M., Hummelshøj, J., & Morozov, D. (2021). Machine learning with persistent homology and chemical word embeddings improves prediction accuracy and interpretability in metal-organic frameworks. *Scientific Reports*, *11*(1), 8888.

Le, T., & Yamada, M. (2018). Persistence fisher kernel: A riemannian manifold kernel for persistence diagrams. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/959ab9a0695c467e7caf75431a872e5c-Paper.pdf

Robinson, A., & Turner, K. (2017). Hypothesis testing for topological data analysis. *Journal of Applied and Computational Topology*, *1*.

Singh, Y., Farrelly, C. M., Hathaway, Q. A., Leiner, T., Jagtap, J., Carlsson, G. E., & Erickson, B. J. (2023). Topological data analysis in medical imaging: Current state of the art. *Insights into Imaging*, *14*(1), 58. https://doi.org/10.1186/s13244-023-01413-w

Wadhwa, R., Dhawan, A., Williamson, D., & Scott, J. (2019). *Tdastats: Pipeline for topological data analysis* [R package version 0.4.1]. https://github.com/rrrlw/TDAstats

Wadhwa, R., Williamson, D. F. K., Dhawan, A., & Scott, J. G. (2018). Tdastats: R pipeline for computing persistent homology in topological data analysis. *Journal of Open Source Software*, *3*(28), 860. https://doi.org/10.21105/joss.00860

Wickham, H. (2011). Testthat: Get started with testing. *The R Journal*, *3*, 5–10. https://doi.org/10.32614/rj-2011-002

Wickham, H., Hester, J., Chang, W., & Bryan, J. (2021). *Devtools: Tools to make developing r packages easier* [R package version 2.4.3]. https://CRAN.R-project.org/package=devtools

Zomorodian, A., & Carlsson, G. (2005). Computing persistent homology. *Discrete and Computational Geometry*, *33*, 249–274. https://doi.org/10.1007/s00454-004-1146-y

2.6 `TDApplied` theory and practice

2.6.1 Introduction

Topological data analysis is a relatively new area of data science which can compare and contrast data sets via non-linear global structure. The main tool of topological data analysis, *persistent homology* (Edelsbrunner, Letscher, and Zomorodian 2000; Zomorodian and G. Carlsson 2005), builds on techniques from the field of algebraic topology to describe shape features present in a data set (stored in a "persistence diagram"). Persistent homology has been used in a number of applications, including

- predicting stock market crashes from the topology of stock price correlations over time (Yen and Cheong 2021),
- finding non-trivial and complex topological structure in local patches of naturalistic images (G. E. Carlsson et al. 2007),
- translating sentences in one language into another language (from a set of candidate sentences) using the persistence diagrams of their word embeddings (Haim Meirom and Bobrowski 2022),
- improving model predictions of various chemical properties of molecules by including topological features (Krishnapriyan et al. 2021), and
- distinguishing between the topology of human brain function of healthy control subjects vs. subjects with a neurological disorder (Gracia-Tabuenca et al. 2020; Hyekyoung et al. 2014; Lee et al. 2011), etc.

For a broad introduction to the mathematical background and main tools of topological data analysis, see Chazal and Michel 2017.

Traditional data science pipelines in academia and industry are focused on machine learning and statistical inference of structured (tabular) data, being able to answer questions like:

- How well can a label variable be predicted from feature variables?

- What are the latent subgroups/dimensions of a dataset?
- Are the subgroups of a dataset, defined by factor features, distinguishable?

While persistence diagrams have been found to be a useful summary of datasets in many domains, they are not structured data and therefore require special analysis methods. Some papers (for example Robinson and Turner 2017; Le and Yamada 2018) have described post-processing pipelines for analyzing persistence diagrams built on distance (Kerber, Morozov, and Nigmetov 2017) and kernel (Le and Yamada 2018) calculations, however these papers are lacking publicly available implementations in R (and Python), and many more data science methods are possible using such calculations (Murphy 2012; Scholkopf, Smola, and Muller 1998; Gretton et al. 2007; M. A. A. Cox and T. F. Cox 2008; Dhillon, Guan, and Kulis 2004).

`TDApplied` is the first R package which provides applied analysis implementations of published methods for analyzing persistence diagrams using machine learning and statistical inference. Its functions contain highly optimized and scalable code (see the package vignette "Benchmarking and speedups") and have been tested and validated (see the package vignette "Comparing distance calculations"). `TDApplied` can interface with other data science packages to perform powerful and flexible analyses (see the package vignette "Personalized analyses with `TDApplied`"), and an example usage of `TDApplied` on real data has been demonstrated (see the package vignette "Human Connectome Project Analysis").

This vignette documents the background of `TDApplied` functions and the usage of those functions on simulated data, by considering a typical data analysis workflow for topological data analysis:

1. Computing and comparing persistence diagrams.
2. Visualizing and interpreting persistence diagrams.
3. Analyzing statistical properties of groups of persistence diagrams.
4. Finding latent structure in groups of persistence diagrams.

5. Predicting labels from persistence diagram structure.

To start we must load the `TDApplied` package:

```
library("TDApplied")
```

Let's get started!

2.6.2 Computing and comparing persistence diagrams

2.6.2.1 Computing diagrams and `TDApplied`'s `PyH` function

The main tool of topological data analysis is called *persistent homology* (Edelsbrunner, Letscher, and Zomorodian 2000; Zomorodian and G. Carlsson 2005). Persistent homology starts with either data points and a distance function, or a distance matrix storing distance values between data points. It assumes that these points arise from a dataset with some kind of "shape." This "shape" has certain features that exist at various scales, but sampling induces noise in these features. Persistent homology aims to describe certain mathematical features of this underlying shape, by forming approximations to the shape at various distance scales. The mathematical features which are tracked include clusters (connected components), loops (ellipses) and voids (spheres), which are examples of *cycles* (i.e. different types of holes). The *homological dimension* of these features are 0, 1 and 2, respectively. What is interesting about these particular mathematical features is that they can tell us where our data is not, which is extremely important information which other data analysis methods cannot provide.

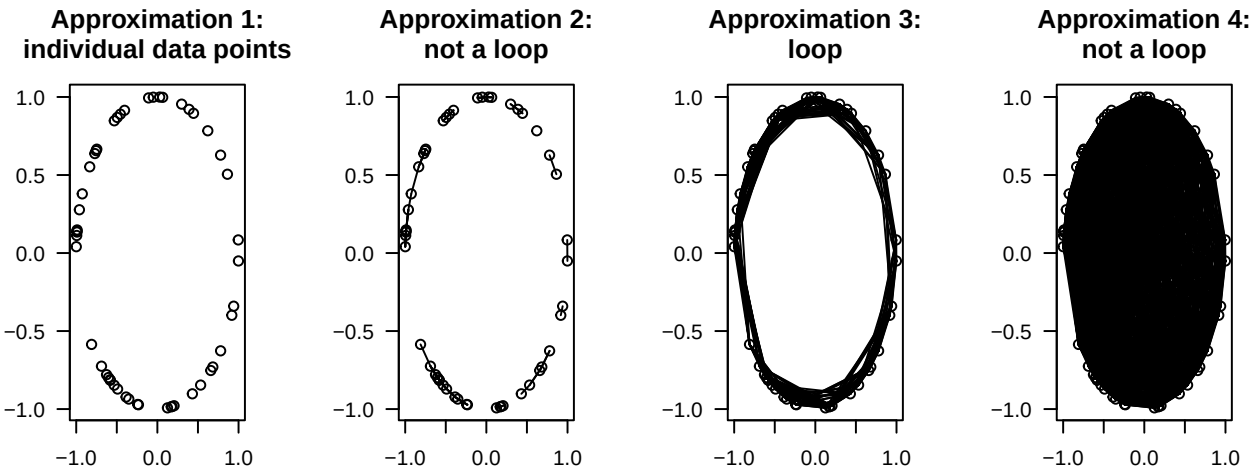

Figure 2.2: An example workflow of persistent homology, noting the linkage radii where a loop exists/does not exist.

The persistent homology algorithm proceeds in the following manner: first, if the input is a dataset and distance metric, then the distance matrix, storing the distance metric value of each pair of points in the dataset, is computed. Next, a parameter $\epsilon \geq 0$ is grown starting at 0, and at each ϵ value we compute a shape approximation of the dataset VR_ϵ, called a *simplicial complex* or in this case a *Vietoris Rips complex*. We construct VR_ϵ by connecting all pairs of points whose distance is at most ϵ. To encode higher-dimensional structure in these approximations, we also add a triangle between any triple of points which are all connected (note that no triangles are formally shaded on the above diagram, even though there are certainly triples of connected points), a tetrahedron between any quadruple of points which are all connected, etc. Note that this process of forming a sequence of skeletal approximations is called a *Rips-Vietoris filtration*, and other methods exist for forming the approximations.

At any given ϵ value, some topological features will exist in VR_ϵ. As ϵ grows, the VR_ϵ's will contain each other, i.e. if $\epsilon_1 < \epsilon_2$, then every edge (triangle, tetrahedron etc.) in VR_{ϵ_1} will also be present in VR_{ϵ_2}. Each topological feature of interest will be "born" at some ϵ_{birth} value, and "die" at some some ϵ_{death} value – certainly each feature will die once the

whole dataset is connected and has trivial shape structure. Consider the example of a loop – a loop will be "born" when the last connection around the circumference of the loop is connected (at the ϵ value which is the largest distance between consecutive points around the loop), and the loop will "die" when enough connections across the loop fill in its hole. Since the topological features are tracked across multiple scales, we can estimate their (static) location in the data, i.e. finding the points on these structures, by calculating what are called *representative cycles*.

The output of persistent homology, a *persistence diagram*, has one 2D point for each topological feature found in the filtration process in each desired homological dimension, where the x-value of the point is the birth ϵ-value and the y-value is the death ϵ-value. Hence every point in a persistence diagram lies above the diagonal – features die after they are born! The difference of a points y and x value, $y - x$, is called the "persistence" of the corresponding topological feature. Points which have high (large) persistence likely represent real topological features of the dataset, whereas points with low persistence likely represent topological noise.

For a more practical and scalable computation of persistence diagrams, a method has been introduced called *persistence cohomology* (Silva, Morozov, and Vejdemo-Johansson 2011; De Silva, Morozov, and Vejdemo-Johansson 2011) which calculates the exact same output as persistent homology (with analogous "representative co-cycles" to persistent homology's representative cycles) just much faster. Persistent cohomology is implemented in the C++ library `ripser` (Bauer 2015), which is wrapped in R in the `TDAstats` package (Wadhwa et al. 2019) and in Python in the `ripser` module. However, it was observed in simulations that the Python implementation of `ripser` seemed faster, even when called in R via the `reticulate` package (Ushey, Allaire, and Tang 2022) (see the package vignette "Benchmarking and speedups"). Therefore, the `TDAstats` `PyH` function has been implemented as a wrapper of the Python `ripser` module for fast calculations of persistence diagrams.

There are three prerequisites that must be satisfied in order to use the `PyH` function:

1. The `reticulate` package must be installed.
2. Python must be installed and configured to work with reticulate.
3. The `ripser` Python module must be installed, via `reticulate::py_install("ripser")`. Some windows machines have had issues with recent versions of the `ripser` module, but version 0.6.1 has been tested and does work on Windows. So, Windows users may try `reticulate::py_install("ripser==0.6.1")`.

For a sample use of `PyH` we will use the following pre-loaded dataset called "circ" (which is stored as a data frame in this vignette):

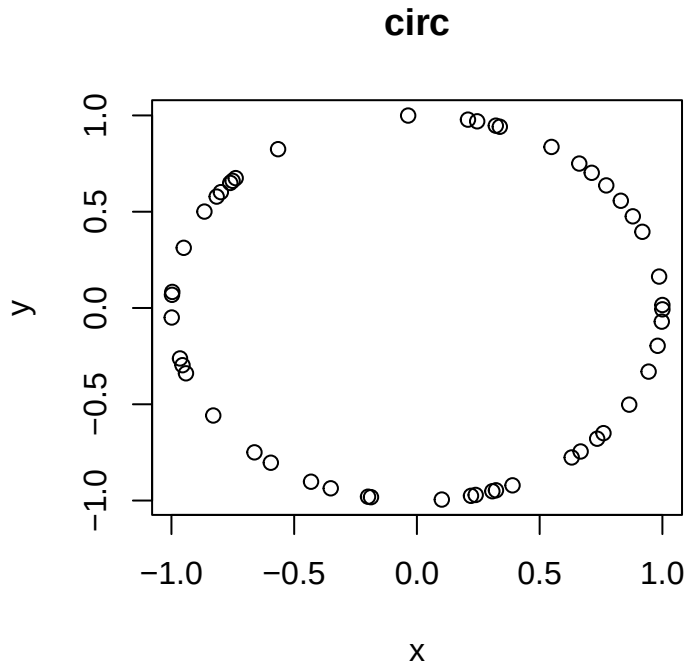

Figure 2.3: The example circ dataset.

We would then calculate the persistence diagram as follows:

```
# import the ripser module
ripser <- import_ripser()

# calculate the persistence diagram
diag <- PyH(X = circ,maxdim = 1,thresh = 2,ripser = ripser)

# view last five rows of the diagram
diag[47:51,]
```

```
#>    dimension     birth     death
#> 47         0 0.0000000 0.2545522
#> 48         0 0.0000000 0.2813237
#> 49         0 0.0000000 0.2887432
#> 50         0 0.0000000 2.0000000
#> 51         1 0.5579783 1.7385925
```

In the package vignette "Benchmarking and speedups," simulations are used to demonstrate the practical advantages of using `PyH` to calculate persistence diagrams compared to other alternatives.

Note that the installation status of Python for `PyH` is checked using the function `reticulate::py_available()`, which according to several online forums does not always behave as expected. If error messages occur using `TDApplied` functions regarding Python not being installed then we recommend consulting online resources to ensure that the `py_available` function returns `TRUE` on your system. Due to the complicated dependencies required to use the `PyH` function, it is only an optional function in the `TDApplied` package and therefore the `reticulate` package is only suggested in the `TDApplied` namespace.

2.6.2.2 Converting diagrams to dataframes with `TDApplied`'s `diagram_to_df` function

The most typical data structure used in R for data science is a data frame. The output of the `PyH` function is a data frame (unless representatives are calculated, in which case the output is a list containing a data frame and other information), but the persistence diagrams calculated from the popular R packages `TDA` (B. T. Fasy et al. 2021) and `TDAstats` (Wadhwa et al. 2019) are not stored in data frames, making subsequent machine learning and inference analyses of these diagrams challenging. Since In order to solve this problem the `TDApplied` function `diagram_to_df` can convert `TDA`/`TDAstats` persistence diagrams into data frames:

```
# convert TDA diagram into data frame
diag1 <- TDA::ripsDiag(circ,maxdimension = 1,maxscale = 2,
                       library = "dionysus")
diag1_df <- diagram_to_df(diag1)
class(diag1_df)
```

```
#> [1] "data.frame"
```

```
# convert TDAstats diagram into data frame
diag2 <- TDAstats::calculate_homology(circ,dim = 1,threshold = 2)
diag2_df <- diagram_to_df(diag1)
class(diag2_df)
```

```
#> [1] "data.frame"
```

When a persistence diagram is calculated with either `PyH`, `ripsDiag` or `alphaComplexDiag` and contains representatives, `diagram_to_df` only returns the persistence diagram data frame (i.e. the representatives are ignored).

2.6.2.3 Comparing persistence diagrams and `TDApplied`'s `diagram_distance` and `diagram_kernel` functions

Earlier we mentioned that persistence diagrams do not form structured data, and now we will give an intuitive argument for why this is the case. A persistence diagram $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ containing n topological features can be represented in a vector of length $2n$, $(x_1, y_1, x_2, y_2, \ldots, x_n, y_n)$. However, we cannot easily combine vectors calculated in this way into a table with a fixed number of feature columns because

(1) persistence diagrams can contain different numbers of features, meaning their vectors would be of different lengths, and
(2) the ordering of the features is arbitrary, calling into question what the right way to compare the vectors would be.

Fortunately, we can still apply many machine learning and inference techniques to persistence diagrams provided we can quantify how near (similar) or far (distant) they are from each other, and these calculations are possible with distance and kernel functions.

There are several ways to compute distances between persistence diagrams in the same homological dimension. The most common two are called the *2-wasserstein* and *bottleneck* distances (Kerber, Morozov, and Nigmetov 2017; Edelsbrunner and Harer 2010). These techniques find an optimal matching of the 2D points in their input two diagrams, and compute a cost of that optimal matching. A point from one diagram is allowed either to be paired (matched) with a point in the other diagram or its diagonal projection, i.e. the nearest point on the diagonal line $y = x$ (matching a point to its diagonal projection is essentially saying that feature is likely topological noise because it died very soon after it was born).

Allowing points to be paired with their diagonal projections both allows for matchings of persistence diagrams with different numbers of points (which is almost always the case in practice) and also formalizes the idea that some points in a persistence diagram represent

noise. The "cost" value associated with a matching is given by either (i) the maximum of infinity-norm distances between paired points, or (ii) the square-root of the sum of squared infinity-norm between matched points. The cost of the optimal matching under loss (i) is the bottleneck distance of persistence diagrams, and the cost of the optimal matching of cost (ii) is called the 2-wasserstein metric of persistence diagrams. Both distance metrics have been used in a number of applications, but the 2-wasserstein metric is able to find more fine-scale differences in persistence diagrams compared to the bottleneck distance. The problem of finding an optimal matching can be solved with the Hungarian algorithm, which is implemented in the R package `clue` (Hornik 2005).

We will introduce three new simple persistence diagrams, D1, D2 and D3, for examples in this section (and future ones):

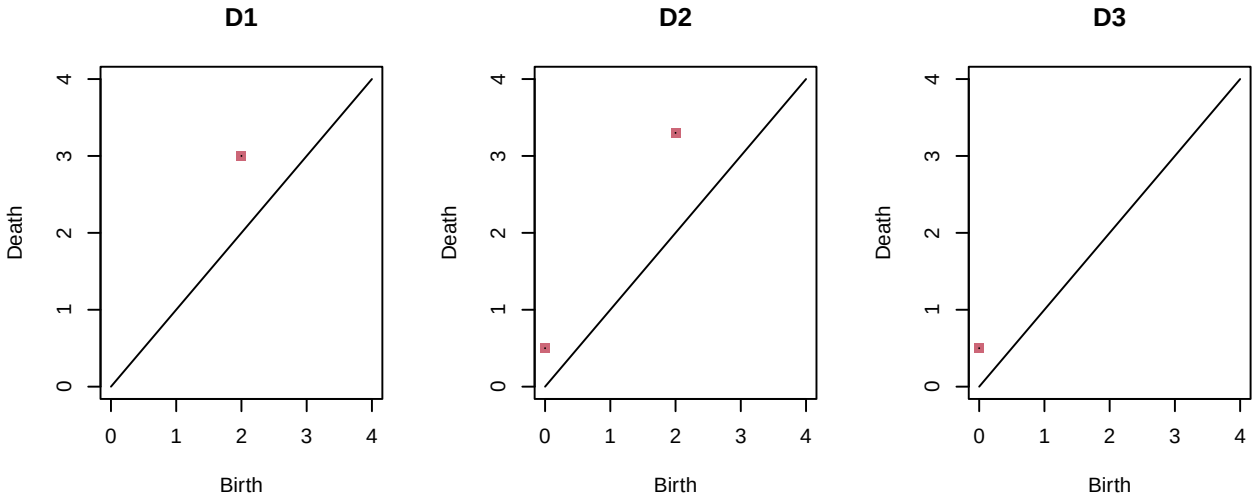

Figure 2.4: Three sample diagrams, D1, D2 and D3, each with one or two 0-dimensional topological features.

Here is a plot of the optimal matchings between D1 and D2, and between D1 and D3:

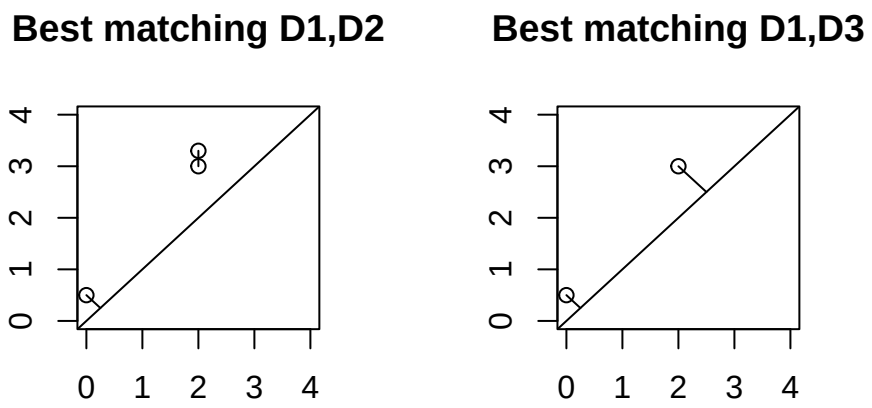

Figure 2.5: The optimal matchings between D1 and D2 (left) and D1 and D3 (right). In the latter matching, each off diagonal point is paired with its own diagonal projection rather than being matched with each other.

In the picture we can see that there is a "better" matching between D1 and D2 compared to D1 and D3, so the (wasserstein/bottleneck) distance value between D1 and D2 would be smaller than that of D1 and D3.

Another distance metric between persistence diagrams, which will be useful for kernel calculations, is called the *Fisher information metric*, $d_{FIM}(D_1, D_2, \sigma)$ (Le and Yamada 2018). The idea is to represent the two persistence diagrams as probability density functions, with a 2D-Gaussian point mass centered at each point in both diagrams (including the diagonal projections of the points in the opposite diagram), all of variance $\sigma^2 > 0$, and calculate how much those distributions disagree on their pdf value at each point in the plane (called their *Fisher information metric*).

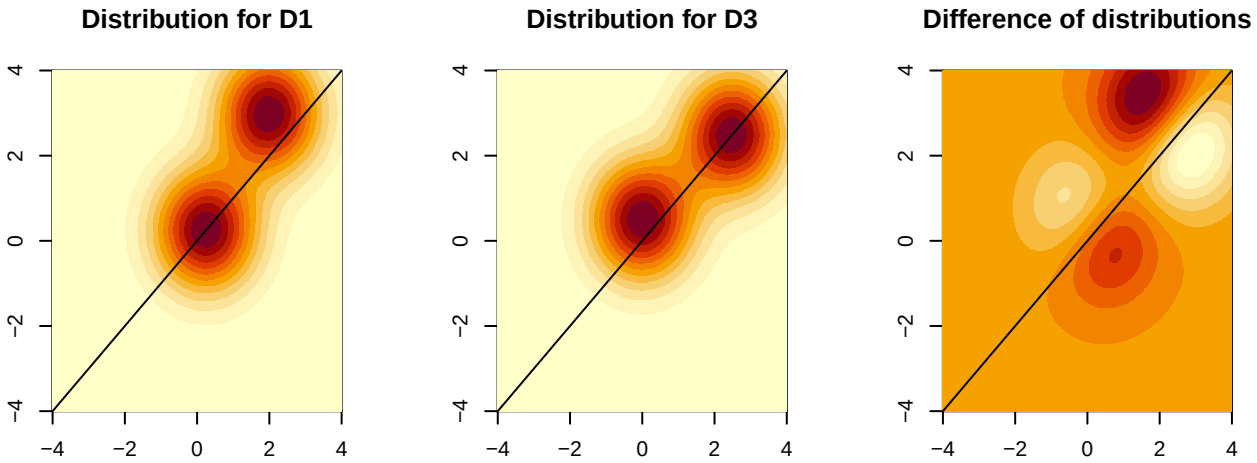

Figure 2.6: Probability distributions which are sums of Gaussian point masses for D1 (left), D3 (center) and the difference of these (right).

Points in the rightmost plot which are close to white in color have the most similar pdf values in the two distributions, and would not contribute to a large distance value; however, having more points with a red color would contribute to a larger distance value.

The wasserstein and bottleneck distances have been implemented in the `TDApplied` function `diagram_distance`. We can confirm that the distance between D1 and D2 is smaller than D1 and D3 for both distances:

```
# calculate 2-wasserstein distance between D1 and D2
diagram_distance(D1,D2,dim = 0,p = 2,distance = "wasserstein")
#> [1] 0.3905125

# calculate 2-wasserstein distance between D1 and D3
diagram_distance(D1,D3,dim = 0,p = 2,distance = "wasserstein")
#> [1] 0.559017

# calculate bottleneck distance between D1 and D2
diagram_distance(D1,D2,dim = 0,p = Inf,distance = "wasserstein")
```

```
#> [1] 0.3


# calculate bottleneck distance between D1 and D3
diagram_distance(D1,D3,dim = 0,p = Inf,distance = "wasserstein")
#> [1] 0.5
```

There is a generalization of the 2-wasserstein distance for any $p \geq 1$, the *p-wasserstein* distance, which can also be computed using the `diagram_distance` function by varying the parameter `p`, although $p = 2$ seems to be the most popular parameter choice.

The `diagram_distance` function can also calculate the Fisher information metric between persistence diagrams:

```
# Fisher information metric calculation between D1 and D2 for
# sigma = 1
diagram_distance(D1,D2,dim = 0,distance = "fisher",sigma = 1)
#> [1] 0.02354779


# Fisher information metric calculation between D1 and D3 for
# sigma = 1
diagram_distance(D1,D3,dim = 0,distance = "fisher",sigma = 1)
#> [1] 0.08821907
```

Again, D1 and D2 are less different than D1 and D3 using the Fisher information metric.

A fast approximation to the Fisher information metric was described in Le and Yamada 2018, and C++ code in the GitHub repository (https://github.com/vmorariu/figtree) was used to calculate this approximation in Matlab. Using the `Rcpp` package (Eddelbuettel and Francois 2011) this code is included in `TDApplied` and the approximation can be calculated by providing the positive `rho` parameter:

```
# Fisher information metric calculation between D1 and D2 for
# sigma = 1
diagram_distance(D1,D2,dim = 0,distance = "fisher",sigma = 1)
```

```
#> [1] 0.02354779
```

```
# fast approximate Fisher information metric calculation between
# D1 and D3 for sigma = 1
diagram_distance(D1,D2,dim = 0,distance = "fisher",sigma = 1,
                 rho = 0.001)
```

```
#> [1] 0.02354779
```

Now we will explore calculating similarity of persistence diagrams using kernel functions. A kernel function is a special (positive semi-definite) symmetric similarity measure between objects in some complicated space which can be used to project data into a space suitable for machine learning (Murphy 2012). Some examples of machine learning techniques which can be "kernelized" when dealing with complicated data are *k-means* (kernel k-means), *principal components analysis* (kernel PCA), and *support vector machines* (SVM) which are inherently based on kernel calculations.

There have been, to date, four main kernels proposed for persistence diagrams. In `TDApplied` the persistence Fisher kernel (Le and Yamada 2018) has been implemented because of its practical advantages over the other kernels – smaller cross-validation SVM error on a number of test data sets and a faster method for cross validation. For information on the other three kernels see Kusano, Fukumizu, and Hiraoka 2018; Carriere, Cuturi, and Oudot 2017; Reininghaus et al. 2015.

The persistence Fisher kernel is computed directly from the Fisher information metric between two persistence diagrams: let $\sigma > 0$ be the parameter for d_{FIM}, and let $t > 0$. Then the persistence Fisher kernel is defined as $k_{PF}(D_1, D_2) = \exp(-t * d_{FIM}(D_1, D_2, \sigma))$.

Computing the persistence Fisher kernel can be achieved with the `diagram_kernel` function in `TDApplied`:

```
# calculate the kernel value between D1 and D2 with sigma, t = 2
diagram_kernel(D1,D2,dim = 0,sigma = 2,t = 2)
#> [1] 0.9872455
# calculate the kernel value between D1 and D3 with sigma, t = 2
diagram_kernel(D1,D3,dim = 0,sigma = 2,t = 2)
#> [1] 0.9707209
```

As before, D1 and D2 are more similar than D1 and D3, and if desired a fast approximation to the kernel value can be computed.

2.6.3 Visualizing and interpreting persistence diagrams

2.6.3.1 `TDApplied's` function `plot_diagram`

Persistence diagrams can contain any number of points representing different types of topological features from different homological dimensions. However we can easily view this information in a two-dimensional plot of the birth and death values of the topological features, where each homological dimension has a unique point shape and color, using `TDApplied's` `plot_diagram` function:

```
plot_diagram(diag,title = "Circle diagram")
```

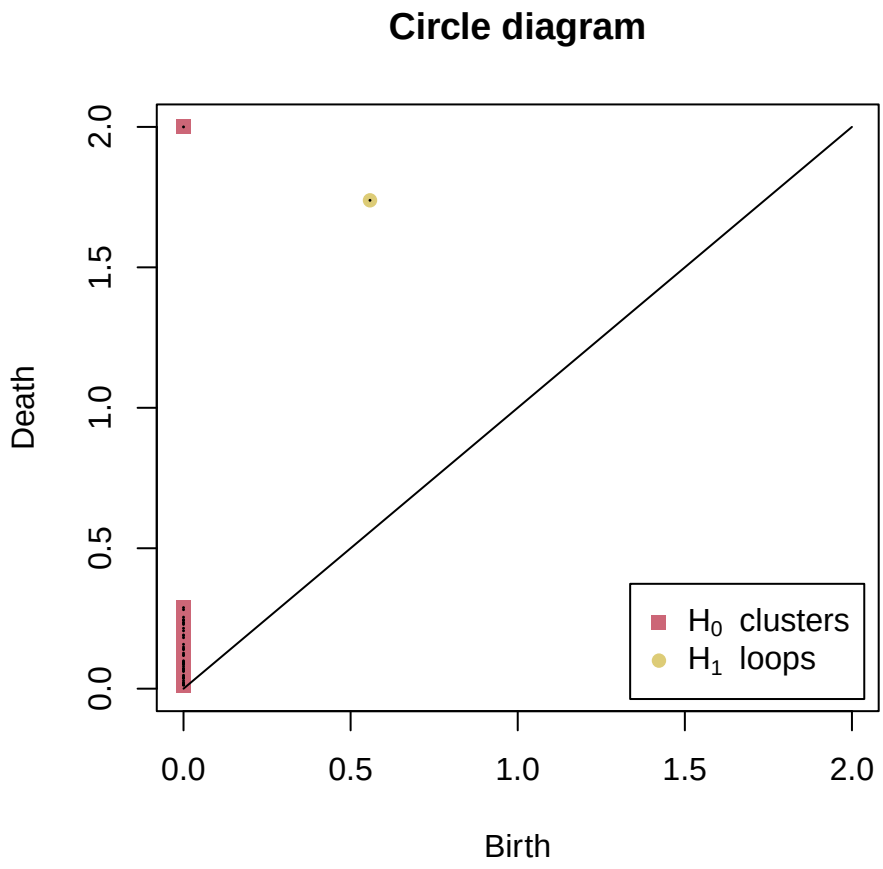

Figure 2.7: The plotted persistence diagram for the circ dataset.

Now that we can visualize persistence diagrams we will describe three tools which can be used for their interpretation – filtering out noisy topological features with bootstrapping, representative (co)cycles and *Vietoris-Rips graphs* (called VR graphs for short).

2.6.3.2 Bootstrapping topological features and `TDApplied`'s `bootstrap_persistence_thresholds` function

Noise in datasets can drastically affect the results of inference and machine learning procedures. Therefore it is desirable to clean data before applying such procedures. Noise in persistence diagrams are low persistence topological features, i.e. features whose birth and death values are very similar (such points would be near the diagonal line when plotted).

The question of determining which points in a persistence diagrams are "significant" and which are "noise" has been addressed via a bootstrapping approach in B. Fasy et al. 2014.

The idea of the procedure is as follows, where X is the input data set and α is the desired threshold for type 1 error:

1. We first compute D, the diagram of X.
2. Then we repeatedly sample, with replacement, the original data to obtain $\{X_1, \ldots, X_n\}$ and compute new persistence diagrams $\{D_1, \ldots, D_n\}$.
3. We calculate the bottleneck distance of each new diagram with the original, $d_\infty(D_i, D)$, in each dimension separately.
4. Finally we compute the $1 - \alpha$ percentile of these values in each dimension.

These thresholds form a square-shaped "confidence interval" around each point in D. In particular, if t was the threshold found for dimension k then the confidence interval around a point $(x, y) \in D$ (of dimension k) is the set of points $\{(x', y') : \max(|x - x'|, |y - y'|) < t\}$. For example, if we calculated the bottleneck-threshold-based confidence interval around the single 1-dimensional point in the circ dataset's persistence diagram, we would get something like this:

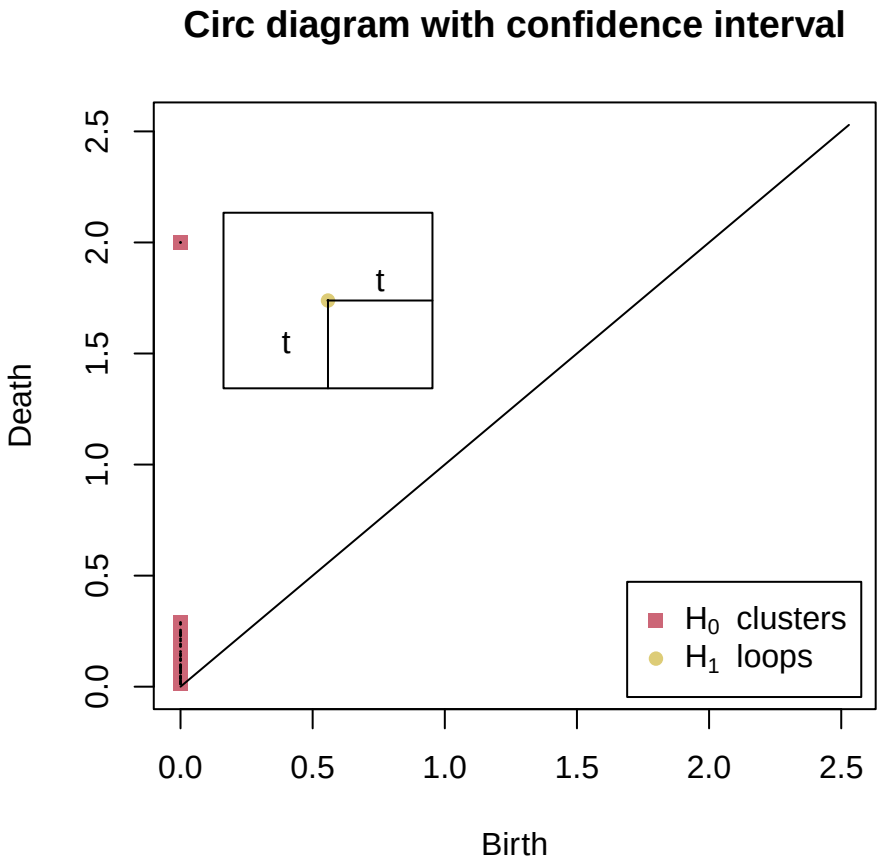

Figure 2.8: An example confidence interval centered at the loop point in the diagram of the circ dataset.

In this setup, "significant" points will be those whose confidence intervals do not touch the diagonal line, i.e. where birth and death is the same. Note that the persistence threshold is twice the bottleneck distance threshold calculated above: the (Euclidean) distance from the point to the bottom right corner of the box is $\sqrt{2}t$, which must be greater than or equal to the (Euclidean) distance of the point to its diagonal projection, which is $\frac{y-x}{\sqrt{2}}$. Therefore, for the point to be considered real (i.e. significant), $\sqrt{2}t \leq \frac{y-x}{\sqrt{2}}$, implying that the persistence of the point, $y - x$, must be no less than twice the bottleneck threshold t.

Like in Robinson and Turner 2017 we can calculate the p-value for a feature as $p = \frac{Z+1}{N+1}$ where Z is the number of bootstrap iterations which, when doubled, are at least the persistence of the feature, and N is the number of bootstrap samples. In order to ensure

that the p-value of any topological feature which survives the thresholding is less than α we transform α by $\alpha \leftarrow \frac{\max\{\alpha(N+1)-1,0\}}{N+1}$.

The function `bootstrap_persistence_thresholds` can be used to determine these persistence thresholds. Here is an example for the circ dataset, and the results can be plotted with the `plot_diagram` function (with overlaid p-values using the graphics `text` function):

```
# calculate the bootstrapped persistence thresholds using 2 cores
# and 30 iterations. We'll use the distance matrix of circ to
# make representative cycles more comprehensible
library("TDA")
thresh <- bootstrap_persistence_thresholds(
X = as.matrix(dist(circ)),
FUN_diag = 'ripsDiag',
FUN_boot = 'ripsDiag',
distance_mat = T,
maxdim = 1,thresh = 2,num_workers = 2,
alpha = 0.05,num_samples = 30,
return_subsetted = T,return_pvals = T,
calculate_representatives = T)
diag <- thresh$diag

# plot original diagram and thresholded diagram side-by-side,
# including p-values. These p-values are the smallest possible
# (1/31) when there are 30 bootstrap iterations
par(mfrow = c(1,2))

plot_diagram(diag,title = "Circ diagram")
```

```
plot_diagram(diag,title = "Circ diagram with thresholds",
             thresholds = thresh$thresholds)
text(x = c(0.2,0.5),y = c(2,1.8),
     paste("p = ",round(thresh$pvals,digits = 3)),
     cex = 0.5)
```

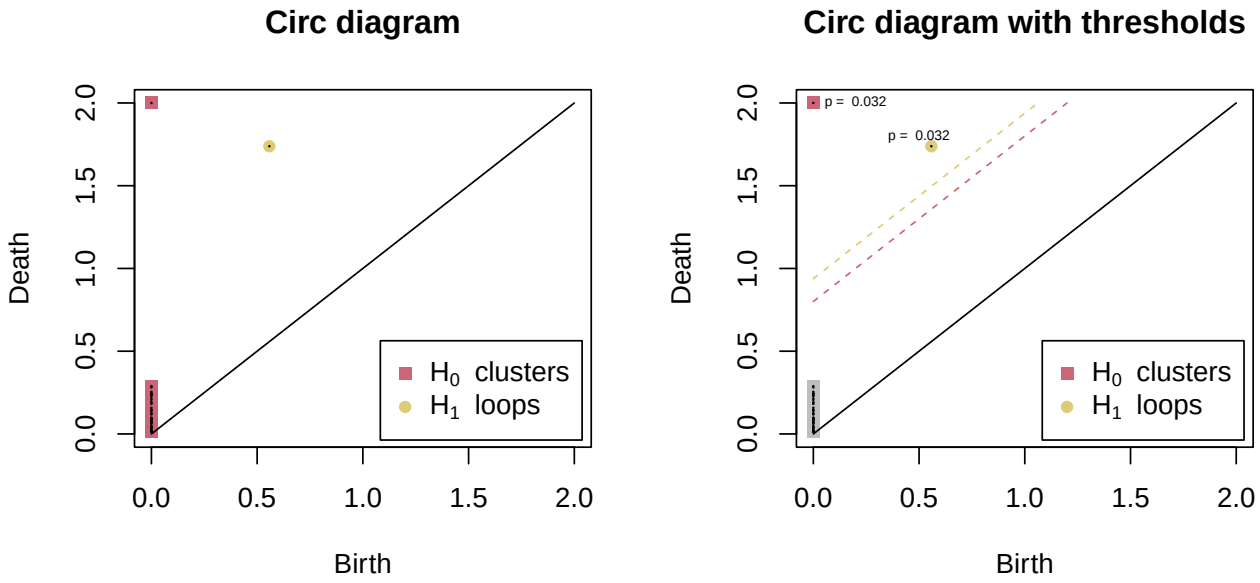

Figure 2.9: The unthresholded diagram of the circ dataset (left) and the same diagram plotted with thresholds and p-values (right). Only one significant component and loop are left after the thresholding procedure.

To see why we needed to load the `TDA` package prior to the function call please see the `bootstrap_persistence_thresholds` function documentation. The bootstrap procedure can be done in parallel or sequentially, depending on which function is specified to calculate the persistence diagrams. There are three possible such functions – `TDAstats`' `calculate_homology`, `TDA`'s `ripsDiag` and `TDApplied`'s `PyH`. The functions `calculate_homology` and `ripsDiag` can be run in parallel. However, `PyH` is the fastest function followed by `calculate_homology` based on our simulations. Both

`ripsDiag` and `PyH` allow for the calculation of representative (co)cycles (i.e. the approximate location in the data of each topological feature), whereas `calculate_homology` does not. Therefore, our recommendation is to pick the function according to the following criteria: if a user can use the `PyH` function, then it should be used in all cases except for when the input data set is small, the machine has many available cores and the number of desired bootstrap iterations is large. Otherwise, use `calculate_homology` for speed or `ripsDiag` for calculating representatives.

2.6.3.3 Representative (co)cycles

One of the advantages of the R package `TDA` over `TDAstats` is its ability to calculate representative cycles in the data, i.e. locating the persistence diagram topological features in the input data set. Having access to representative cycles can permit deep analyses of the original data set by finding particular types of features spanned by certain subsets of data points. For example, a representative cycle of a 1-dimensional topological feature would be a set of edges between data points which lie along that feature (a loop). The `PyH` function can also find *representative cocycles* (i.e. analogues of representative cycles for persistent cohomology) in its input data, which are returned if the `calculate_representatives` parameter is set to `TRUE`. In that case, the `PyH` function returns a list with a data frame called "diagram," containing the persistence diagram, and a list called "representatives." The "representatives" list has one element for each homological dimension in the persistence diagram, with one matrix/array for each point in the persistence diagram of that dimension (except for dimension 0, where the list is always empty). The matrix for a d-dimensional feature (1 for loops, 2 for voids, etc.) has $d + 1$ columns, where row i contains the row indices in the data set of the data points in the i-th substructure in the representative (a substructure of a loop would be an edge, a substructure of a void would be a triangle, etc.). Here is an example where we calculate the representative cocycles of our circ dataset:

```
# ripser has already been imported, so calculate diagram
# with representatives
diag_rep <- PyH(circ,maxdim = 1,thresh = 2,ripser = ripser,
                calculate_representatives = T)

# identify the loops in the diagram
diag_rep$diagram[which(diag_rep$diagram$dimension == 1),]
```

```
#>    dimension     birth    death
#> 50         1 0.5579783 1.738593
```

```
# show the representative for the loop, just the first five rows
diag_rep$representatives[[2]][[1]][1:5,]
```

```
#>      [,1] [,2]
#> [1,]   50   42
#> [2,]   46   42
#> [3,]   50    4
#> [4,]   42   29
#> [5,]   42   16
```

The representative of the one loop contains the edges found to be present in the loop. We could iterate over the representative for the loop to find all the data points in that representative:

```
unique(c(diag_rep$representatives[[2]][[1]][,1],
         diag_rep$representatives[[2]][[1]][,2]))
```

```
#>  [1] 50 46 42 29 16  7 48  4 11 25 40 37 22 43 15 20 34 32 27
#> [20] 8 44 36 39  5  1 21 24
```

Since the circ dataset is two-dimensional, we could actually plot the loop representative according to the following process:

1. Pick a threshold ϵ between the birth and death radii of the cocycle.
2. Plot all edges between pairs of points in circ of distance no more than ϵ.
3. Highlight all edges in the representative.

Since the death radius of the main cocycle is 1.738894, we can use the following code to plot the cocycle at thresholds value 1.7:

```
plot(x = circ$x,y = circ$y,xlab = "x",ylab = "y",
     main = "circ with representative")
for(i in 1:nrow(circ))
{
  for(j in 1:nrow(circ))
  {
    pt1 <- circ[i,]
    pt2 <- circ[j,]
    if(sqrt((pt1[[1]]-pt2[[1]])^2 + (pt1[[2]]-pt2[[2]])^2) <= 1.7)
    {
      graphics::lines(x = c(pt1[[1]],pt2[[1]]),
                      y = c(pt1[[2]],pt2[[2]]))
    }
  }
}

for(i in 1:nrow(diag_rep$representatives[[2]][[1]]))
{
  pt1 <- circ[diag_rep$representatives[[2]][[1]][i,1],]
  pt2 <- circ[diag_rep$representatives[[2]][[1]][i,2],]
```

```
  if(sqrt((pt1[[1]]-pt2[[1]])^2 + (pt1[[2]]-pt2[[2]])^2) <= 1.7)
  {
    graphics::lines(x = c(pt1[[1]],pt2[[1]]),
                    y = c(pt1[[2]],pt2[[2]]),col = "red")
  }
}
```

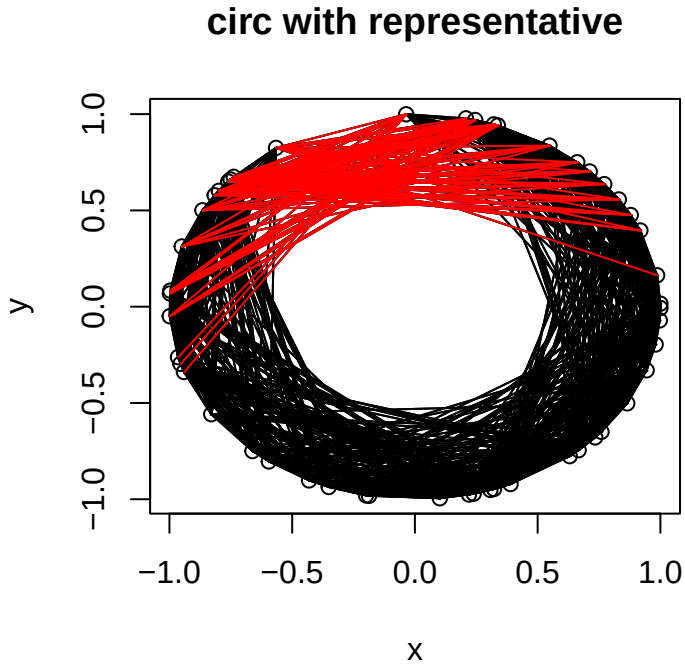

Figure 2.10: The representative cocycle of the loop in the circ dataset, plotted as red edges. If these edges were removed the loop would cease to exist.

This plot shows the main loop that was found via persistent cohomology, and the representative is a set of edges (in this 2D case) whose removal would destroy the loop. A more intuitive notion of a "representative loop" can be found with persistent homology, for instance using the `TDA` `ripsDiag` function with the `location` parameter set to `TRUE`. Another example of the representative cocycle can be found using another threshold value, for instance the (rounded up) birth value of 0.6009:

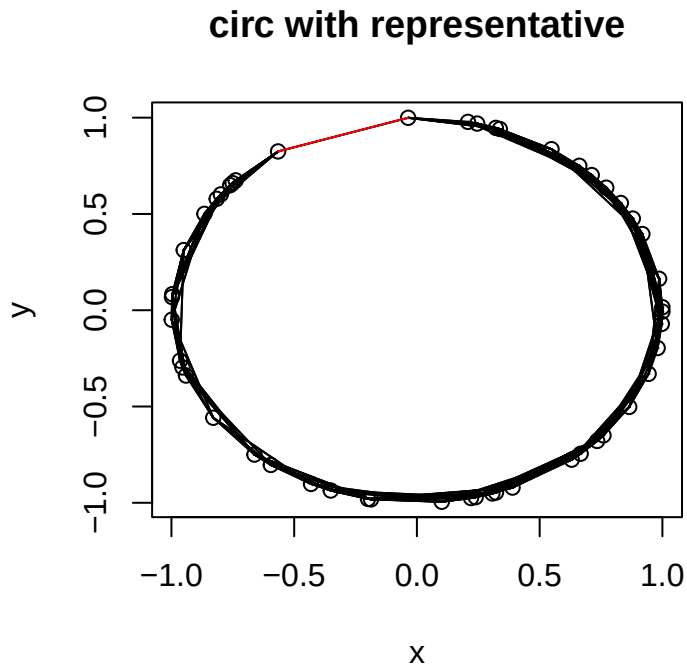

Figure 2.11: A more "minimal" representative cocycle, i.e. with fewer redundant edges around the loop.

Since we know that the data points in the representatives help form a loop in the original data set, we could perform a further exploratory analysis in circ to explore the periodic nature of the feature. While in this example we know that a loop will be present, this type of analysis could help find hidden latent structure in data sets.

2.6.3.4 VR graphs and `TDApplied`'s `vr_graphs` and `plot_vr_graph` functions

Integral to the plots in the previous section were that the circ dataset is 2D, but most datasets are not 2D. In order to investigate topological features of high-dimensional datasets we can use the Vietoris-Rips complexes which are calculated as an intermediate step in persistent homology (as described earlier). Recall that the Vietoris-Rips complex of a dataset is a skeletal representations of the dataset's structure at a particular scale ϵ formed by connecting data points of distance at most ϵ and adding higher-dimensional structures (like triangles between triples of connected points). The connections between

data points in a Vietoris-Rips complex form a "VR graph" (Zomorodian 2010), and this graph can be visualized regardless of the dimension of the underlying dataset.

We can use the function `vr_graphs` to compute VR graphs at a variety of scales, and can visualize the resulting graphs with the `igraph` package (Csardi and Nepusz 2006) and the `plot_vr_graph` function. To demonstrate this functionality, we will pick two ϵ scales to study the dataset structure of circ, only based on its persistence diagram - the first scale will be half the birth radius of the loop and the second scale will be the average of the loop's birth and death radius:

```
# get half of loop's birth radius
eps_1 <- diag[nrow(diag),2L]/2


# get mean of loop's birth and death radii
eps_2 <- (diag[nrow(diag),3L] + diag[nrow(diag),2L])/2


# compute two VR graphs
gs <- vr_graphs(X = circ,eps = c(eps_1,eps_2))
```

Next we can plot both VR graphs:

```
# plot first graph
plot_vr_graph(gs,eps_1)
```

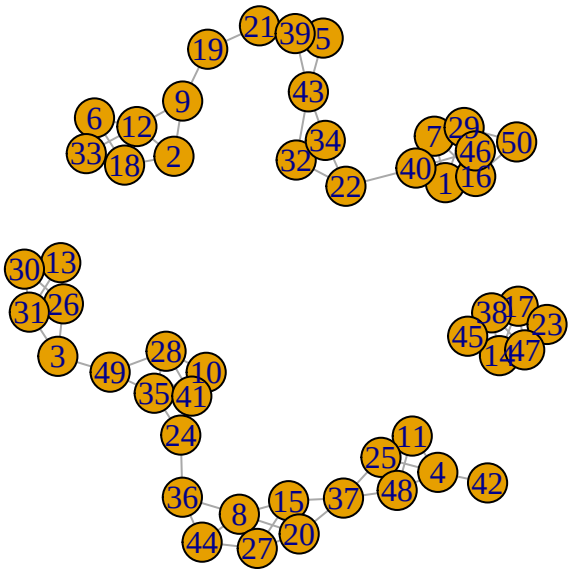

Figure 2.12: The VR graph at the lower ϵ radius, which is a scale at which the loop does not yet exist.

```
# plot second graph
plot_vr_graph(gs,eps_2)
```

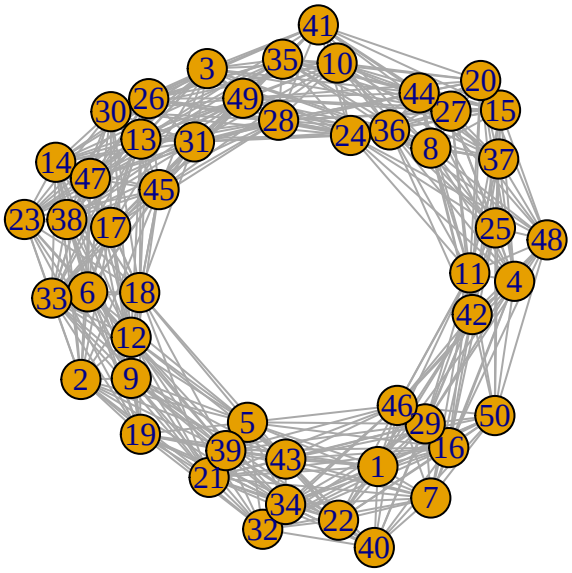

Figure 2.13: The VR graph at the larger ϵ radius, which is a scale at which the loop does exist.

The first graph shows that at a scale before the loop is born, the dominant loop does not exist in the data (and that the space is more fragmented), while the second graph was able to retrieve the dominant loop. Visualizing multiple VR graphs of a dataset, choosing particular scales of interest from the persistence diagram of the dataset, can be an effective tool for interpreting topological features.

The input parameters of `plot_vr_graph` can help customize the graph visualizations. For example, if we want to investigate the loop we can also customize the plot to highlight the loop representative, and to only plot the component of the graph which contains the loop vertices:

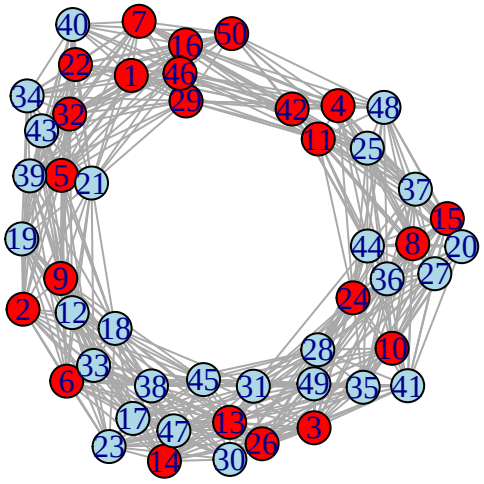

Figure 2.14: A VR graph of the loop with nodes in the representative cycle (computed with persistent homology in the `TDA` package) highlighted in red.

We can customize one level further by removing the vertex labels and using the graph layout (i.e. x-y coordinates of all vertices) to plot other things on the graph:

```
# plot only component containing the loop stimuli with
# vertex colors
layout <- plot_vr_graph(gs,eps_2,cols = colors,
                component_of = stimuli_in_loop[[1]],
```

```
                vertex_labels = F,return_layout = T)
layout <- apply(layout,MARGIN = 2,FUN = function(X){

  return(-1 + 2*(X - min(X))/(max(X) - min(X)))

})

# get indices of vertices in loop
# not necessary in this case but necessary when we have
# removed some vertices from the graph
vertex_inds <- match(stimuli_in_loop,as.numeric(rownames(layout)))

# add volcano image over loop nodes
# image could be anything like rasters read from
# png files! this is just an example..
utils::data("volcano")
volcano <- (volcano - min(volcano)) / diff(range(volcano))
for(i in vertex_inds)
{
  graphics::rasterImage(volcano,xleft = layout[i,1L] - 0.05,
                                xright = layout[i,1L] + 0.05,
                                ybottom = layout[i,2L] - 0.05,
                                ytop = layout[i,2L] + 0.05)
}
```

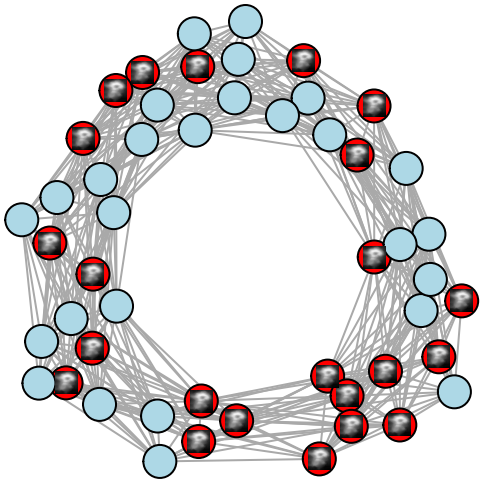

Figure 2.15: The same VR graph as before but with images placed over the nodes in the representative cycle.

These visualizations can help determine which points are part of a representative and what the dataset structure is at various scales.

2.6.4 Hypothesis testing

Having visualized and interpreted our data of interest (persistence diagrams), a common next step in data analysis pipelines is to ask statistical questions in the form of hypothesis testing (Casella, Berger, and Company 2002). Two such questions that we will focus on are:

1. Are groups of persistence diagrams different from each other?
2. Are two groups of persistence diagrams independent of each other?

In order to answer these questions we need to generate groups of persistence diagrams. We will generate diagrams which are random deviations of D1, D2 and D3 diagrams from earlier:

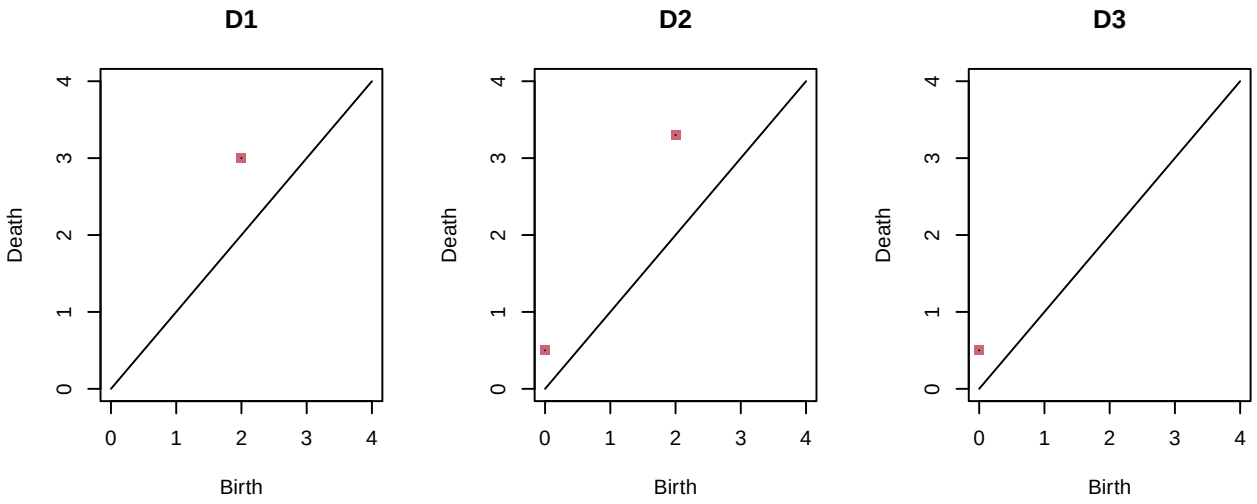

Figure 2.16: The same D1, D2 and D3 persistence diagrams from earlier examples.

The helper function `generate_TDApplied_vignette_data` (seen below) adds Gaussian noise with a small variance to the birth and death values of the points in D1, D2 and D3, making "noisy copies" of the three diagrams:

Here is an example D1 and two noisy copies:

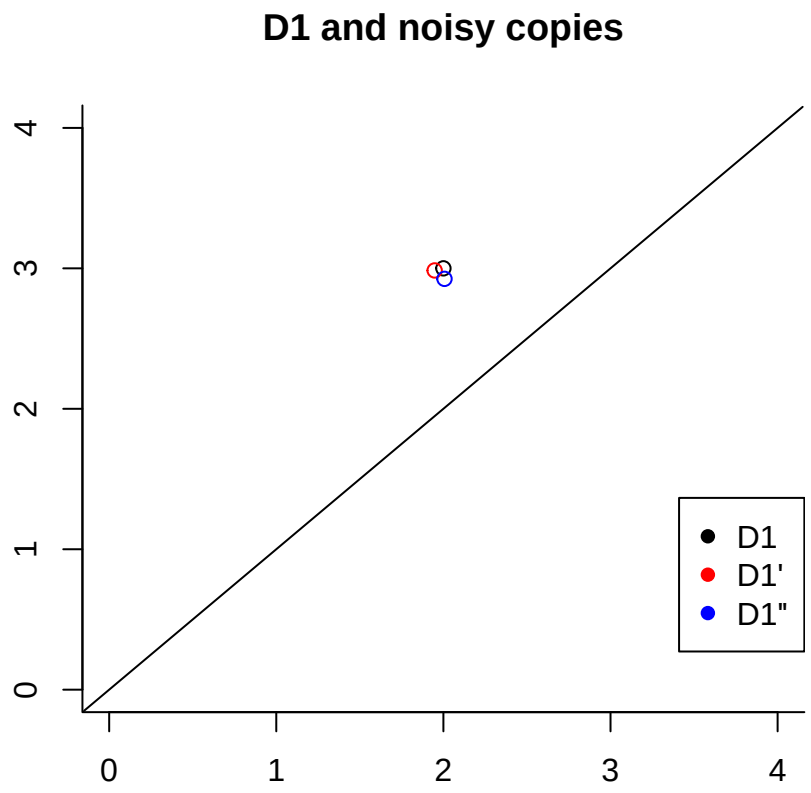

Figure 2.17: D1 and two noisy copies of D1. The single point in D1 is moved randomly by a 2D Gaussian distribution of small variance.

2.6.4.1 Detecting group differences and `TDApplied`'s `permutation_test` function

One of the most important inference procedures in classical statistics is the analysis of variance (ANOVA), which can find differences in the means of groups of normally-distributed measurements (Casella, Berger, and Company 2002). Distributions of persistence diagrams and their means can be complicated (see Turner 2013 and Turner et al. 2014). Therefore, a non-parametric permutation test has been proposed which can find differences in groups of persistence diagrams. Such a test was first proposed in Robinson and Turner 2017, and some variations have been suggested in later publications. In Robinson and Turner

2017, two groups of persistence diagrams would be compared. The null hypothesis, H_0, is that the diagrams from the two groups are generated from shapes with the same type and scale of topological features, i.e. they "come" from the same "shape." The alternative hypothesis, H_A, is that the underlying type or scale of the features are different between the two groups. In each dimension a p-value is computed, finding evidence against H_0 in that dimension. A measure of within-group distances (a "loss function") is calculated for the two groups, and that measure is compared to a null distribution for when the group labels are permuted.

This inference procedure is implemented in the `permutation_test` function, with several speedups and additional functionalities. Firstly, the loss function is computed in parallel for scalability since distance computations can be expensive. Secondly, we store distance calculations as we compute them because these calculations are often repeated. Additional functionality includes allowing for any number groups (not just two) and allowing for a pairing between groups of the same size as described in Abdallah et al. 2023. When a natural pairing exists between the groups (such as if the groups represent persistence diagrams from the same subject of a study in different conditions) we can simulate a more realistic null distribution by restricting the way in which we permute group labels, achieving higher statistical power.

In order to demonstrate the permutation test we will detect differences between noisy copies of D1, D2, D3:

```
# permutation test between three diagrams
g1 <- generate_TDApplied_vignette_data(3,0,0)
g2 <- generate_TDApplied_vignette_data(0,3,0)
g3 <- generate_TDApplied_vignette_data(0,0,3)
perm_test <- permutation_test(g1,g2,g3,
                              num_workers = 2,
                              dims = c(0))
```

```
perm_test$p_values
#>          0
#> 0.04761905
```

As expected, a difference was found (at the $\alpha = 0.05$ significance level) between the three groups.

The package `TDAstats` also has a function called `permutation_test` which is based on the same test procedure. However, it uses an unpublished distance metric between persistence diagrams (see the package vignette "Comparing distance calculations") and does not use parallelization for scalability. As such, care must be taken when both `TDApplied` and `TDAstats` are attached in an R script to use the particular `permutation_test` function desired.

2.6.4.2 Independence between two groups of paired diagrams and `TDApplied`'s `independence_test` function

An important question when presented with two groups of paired data is determining if the pairings are independent or not. A procedure was described in Gretton et al. 2007 which can be used to answer this question using kernel computations, and importantly uses a parametric null distribution. The null hypothesis for this test is that the groups are independent, and the alternative hypothesis is that the groups are not independent. A test statistic called the *Hilbert-Schmidt independence criteria* (Gretton et al. 2007) is calculated, and its value is compared to a gamma distribution with certain parameters which are estimated from the data.

This inference procedure has been implemented in the `independence_test` function, and returns the p-value of the test in each desired dimension of the diagrams (among other additional information). We would expect to find no dependence between noisy copies of D1, D2 and D3, since each copy is generated randomly:

```
# create 10 noisy copies of D1 and D2
g1 <- generate_TDApplied_vignette_data(10,0,0)
g2 <- generate_TDApplied_vignette_data(0,10,0)

# do independence test with sigma = t = 1
indep_test <- independence_test(g1,g2,dims = c(0),num_workers = 2)
indep_test$p_values
#>         0
#> 0.4314036
```

The p-value of this test would not be significant at any typical significance threshold, reflecting the fact that there is no real (i.e. non-spurious) dependence between the two groups, as expected.

2.6.5 Finding latent structure

Patterns may exist in a collection of persistence diagrams. For example, if the collection contained diagrams from three distinct shapes then we would like to be able to assign three distinct (discrete) labels to the correct subsets of diagrams. On the other hand, maybe the diagrams vary along several dimensions, like the mean persistence of their loops and the mean persistence of their components. In this case we would like to be able to retrieve these continuous features for visualizing all diagrams in the same (2D in this example) space. These two types of analyses are called clustering and dimension reduction respectively, and are two of the most common and popular machine learning applications. We will explore three techniques from these areas - kernel k-means from clustering, and multidimensional scaling and kernel principal components analysis from dimension reduction.

2.6.5.1 Kernel k-means and `TDApplied`'s `diagram_kkmeans` function

Kernel k-means (Dhillon, Guan, and Kulis 2004) is a method which can find hidden groups in complex data, extending regular k-means clustering (Murphy 2012) via a kernel. A "kernel distance" is calculated between a persistence diagram and a cluster center using only the kernel function, and the algorithm converges like regular k-means. This algorithm is implemented in the function `diagram_kkmeans` as a wrapper of the `kernlab` (Karatzoglou et al. 2004) function `kkmeans`. Moreover, a prediction function `predict_diagram_kkmeans` can be used to find the nearest cluster labels for a new set of diagrams. Here is an example clustering three groups of noisy copies from D1, D2 and D3:

```
# create noisy copies of D1, D2 and D3
g <- generate_TDApplied_vignette_data(3,3,3)

# calculate kmeans clusters with centers = 3, and sigma = t = 2
clust <- diagram_kkmeans(diagrams = g,centers = 3,dim = 0,t = 2,
                         sigma = 2,num_workers = 2)

# display cluster labels
clust$clustering@.Data
#> [1] 2 2 2 3 3 3 1 1 1
```

As we can see, the `diagram_kkmeans` function was able to correctly separate the three generating diagrams D1, D2 and D3 (the cluster labels are arbitrary and therefore may not be 1,1,1,2,2,2,3,3,3; however, they induce the correct partition).

If we wish to predict the cluster label for new persistence diagrams (computed via the largest kernel value to any cluster center), we can use the `predict_diagram_kkmeans` function as follows:

```
# create nine new diagrams
g_new <- generate_TDApplied_vignette_data(3,3,3)

# predict cluster labels
predict_diagram_kkmeans(new_diagrams = g_new,clustering = clust,
                        num_workers = 2)
#> [1] 2 2 2 3 3 3 1 1 1
```

This function correctly predicted the cluster for each new diagram (assigning each diagram to the cluster label by D1, D2 or D3, depending on which diagram it was generated from).

2.6.5.2 Multidimensional scaling and `TDApplied`'s `diagram_mds` function

Dimension reduction is a task in machine learning which is commonly used for data visualization, removing noise in data, and decreasing the number of covariates in a model (which can be helpful in reducing overfitting). One common dimension reduction technique in machine learning is called *multidimensional scaling* (MDS) (M. A. A. Cox and T. F. Cox 2008). MDS takes as input an n by n distance (or dissimilarity) matrix D, computed from n points in a dataset, and outputs an embedding of those points into a Euclidean space of chosen dimension k which best preserves the inter-point distances. MDS is often used for visualizing data in exploratory analyses, and can be particularly useful when the input data points do not live in a shared Euclidean space (as is the case for persistence diagrams). Using the R function `cmdscale` from the package `stats` (R Core Team 2021) we can compute the optimal embedding of a set of persistence diagrams using any of the three distance metrics with the function `diagram_mds`. Here is an example of the `diagram_mds` function projecting nine persistence diagrams, three noisy copies sampled from each of D1, D2 and D3, into 2D space:

```
# create 9 diagrams based on D1, D2 and D3
g <- generate_TDApplied_vignette_data(3,3,3)

# calculate their 2D MDS embedding in dimension 0 with
# the bottleneck distance
mds <- diagram_mds(diagrams = g,dim = 0,p = Inf,k = 2,
                   num_workers = 2)

# plot
par(mar=c(5.1, 4.1, 4.1, 8.1), xpd=TRUE)
plot(mds[,1],mds[,2],xlab = "Embedding coordinate 1",
     ylab = "Embedding coordinate 2",
     main = "MDS plot",
     col = as.factor(rep(c("D1","D2","D3"),each = 3)),
     bty = "L")
legend("topright", inset=c(-0.2,0),
       legend=levels(as.factor(c("D1","D2","D3"))),
       pch=16, col=unique(as.factor(c("D1","D2","D3"))))
```

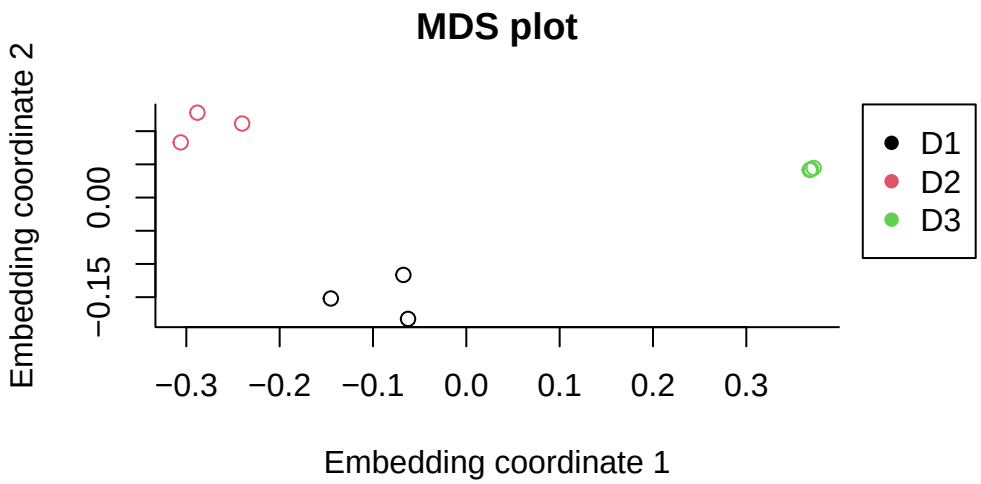

Figure 2.18: MDS plot of the nine persistence diagrams based on the distances between them. The three groups of diagrams are clearly separated.

The MDS plot shows the clear separation between the three generating diagrams (D1, D2 and D3), and the embedded coordinates could be used for further downstream analyses.

2.6.5.3 Kernel principal components analysis, and `TDApplied`'s `diagram_kpca` function

PCA is another dimension reduction technique in machine learning, but can be preferable compared to MDS in certain situations because it allows for the projection of new data points onto an old embedding model (Murphy 2012). For example, this can be important if PCA is used as a pre-processing step in model fitting. Kernel PCA (kPCA) (Scholkopf, Smola, and Muller 1998) is an extension of regular PCA which uses a kernel to project complex data into a high-dimensional Euclidean space and then uses PCA to project that data into a low-dimensional space. The `diagram_kpca` method computes the kPCA embedding of a set of persistence diagrams, and the `predict_diagram_kpca` function can be used to project new diagrams using a pre-trained kPCA model. Here is an example using a group of noisy copies of D1, D2 and D3:

```
# create noisy copies of D1, D2 and D3
g <- generate_TDApplied_vignette_data(3,3,3)

# calculate their 2D PCA embedding with sigma = t = 2
pca <- diagram_kpca(diagrams = g,dim = 0,t = 2,sigma = 2,
                    features = 2,num_workers = 2)

# plot
par(mar=c(5.1, 4.1, 4.1, 8.1), xpd=TRUE)
plot(pca$pca@rotated[,1],pca$pca@rotated[,2],
     xlab = "Embedding coordinate 1",
     ylab = "Embedding coordinate 2",main = "PCA plot",
     col = as.factor(rep(c("D1","D2","D3"),each = 3)))
legend("topright",inset = c(-0.2,0),
       legend=levels(as.factor(c("D1","D2","D3"))), pch=16,
       col=unique(as.factor(c("D1","D2","D3"))))
```

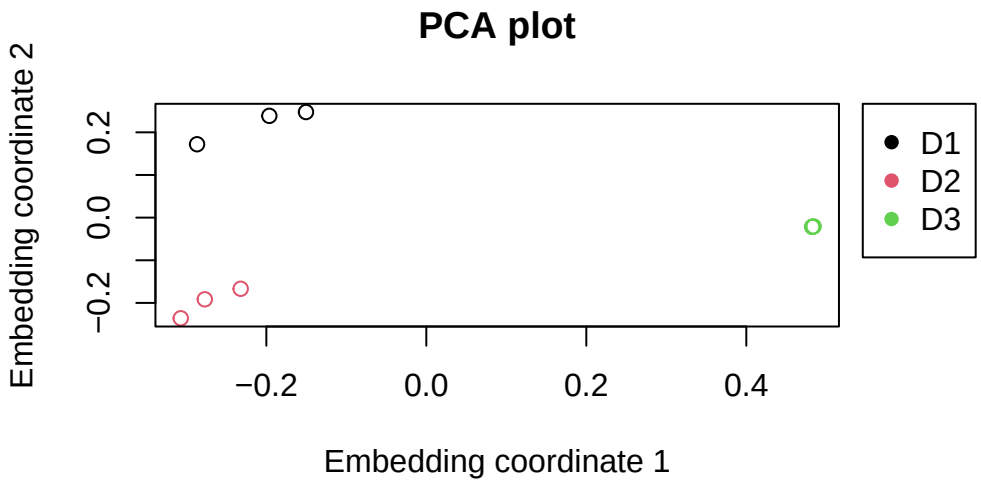

Figure 2.19: Kernel PCA plot of the nine persistence diagrams based on their kernel similarity values. The three groups of diagrams are again clearly separated.

The function was able to recognize the three groups, and the embedding coordinates can be used for further downstream analysis. However, an important advantage of kPCA over MDS is that in kPCA we can project new points onto an old embedding using the `predict_diagram_kpca` function:

```
# create nine new diagrams
g_new <- generate_TDApplied_vignette_data(3,3,3)

# project new diagrams onto old model
new_pca <- predict_diagram_kpca(new_diagrams = g_new,
                                embedding = pca,num_workers = 2)

# plot
par(mar=c(5.1, 4.1, 4.1, 8.1), xpd=TRUE)
plot(new_pca[,1],new_pca[,2],xlab = "Embedding coordinate 1",
     ylab = "Embedding coordinate 2",main = "PCA prediction plot",
     col = as.factor(rep(c("D1","D2","D3"),each = 3)))
legend("topright",inset = c(-0.2,0),
       legend=levels(as.factor(c("D1","D2","D3"))), pch=16,
       col=unique(as.factor(c("D1","D2","D3"))))
```

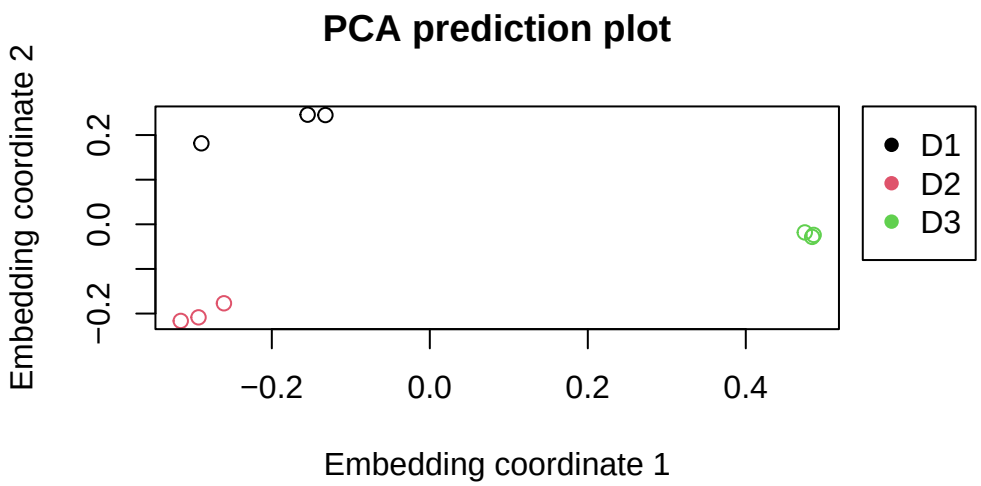

Figure 2.20: Nine new persistence diagrams are projected into the same 2D space of the precomputed kernel PCA model. The three groupings of diagrams maintained both their separation and position in 2D space.

As we can see, the original three groups, and their approximate location in 2D space, is preserved during prediction.

2.6.6 Predicting labels of persistence diagrams

One of the most valuable problems that machine learning is used to solve is that of prediction - can a variable Y be predicted from other variables X. Kernel functions can be used to predict a variable Y from a persistence diagram D using a class of models called support vector machines (SVMs) (Murphy 2012).

2.6.6.1 Support vector machines and `TDApplied`'s `diagram_ksvm` function

SVMs are one of the most popular machine learning techniques for regression and classification tasks. SVMs use a kernel function to project complex data into a high-dimensional space and then find a sparse set of training examples, called "support vectors," which maximally linearly separate the outcome variable classes (or yield the highest explained variance in the case of regression).

Unlike in the case of dimension reduction or clustering, it is possible that our persistence diagrams may each have a label, either discrete or continuous, which gives us more information about the underlying data represented by the diagram. For instance, if our persistence diagrams represented periodic behavior in stock market trends during a particular temporal window then a useful (discrete) label could be whether the overall market went up or down during that period. Being able to predict such labels from the persistence diagrams themselves is a way to link persistence diagrams to external data through modeling, i.e. classification and regression.

SVMs have been implemented in `TDApplied` by the function `diagram_ksvm`, tailored for input datasets which contain pairs of persistence diagrams and their outcome variable labels. A prediction method is supplied called `predict_diagram_ksvm` which can be used to predict the label value of a set of new persistence diagrams given a pre-trained model. A parallelized implementation of cross-validation model-fitting is used based on the remarks in (Le and Yamada 2018) for scalability (which avoids needlessly recomputing persistence Fisher information metric values). Here is an example of fitting an SVM model on a list of persistence diagrams for a classification task (guessing whether the diagram comes from D1, D2 or D3):

```
# create thirty noisy copies of D1, D2 and D3
g <- generate_TDApplied_vignette_data(10,10,10)

# create response vector
y <- as.factor(rep(c("D1","D2","D3"),each = 10))

# fit model with cross validation
model_svm <- diagram_ksvm(diagrams = g,cv = 2,dim = c(0),
                          y = y,sigma = c(1,0.1),t = c(1,2),
                          num_workers = 2)
```

We can use the function `predict_diagram_ksvm` to predict new diagrams like so:

```
# create nine new diagrams
g_new <- generate_TDApplied_vignette_data(3,3,3)


# predict
predict_diagram_ksvm(new_diagrams = g_new,model = model_svm,
                     num_workers = 2)
#> [1] D1 D1 D1 D2 D2 D2 D3 D3 D3
#> Levels: D1 D2 D3
```

As we can see the best SVM model was able to separate the three diagrams We can gain more information about the best model found during model fitting and the CV results by accessing different list elements of `model_svm`.

2.6.7 Limitations of `TDApplied` functionality

There is one main limitation of `TDApplied` which should be discussed for its own future improvements – `TDApplied` functions cannot analyze numeric and factor features with persistence diagrams. This may be too inflexible for some applications, where the data may include several persistence diagrams, or a mix of persistence diagrams, numeric and categorical variables. The package vignette "Personalized analyses with `TDApplied`" demonstrates how one can circumvent this issue using extra code; however, a future update to `TDApplied` might construct such functionality directly into its functions.

2.6.8 Conclusion

Current topological data analysis packages in R (and Python) do not provide the ability to carry out statistics and machine learning with persistence diagrams, leading to a high barrier to adoption of topological data analysis in academia and industry. By filling in this gap, the `TDApplied` package aims to bridge topological data analysis with researchers

and data practitioners in the R community. Topological data analysis is an exciting and powerful new field of data analysis, and with `TDApplied` anyone can access its power for meaningful and creative analyses of data.

2.6.9 References

Abdallah, Hassan et al. (2023). "Statistical inference for persistent homology applied to simulated fMRI time series data". In: *Foundations of Data Science* 5.1, pp. 1–25. DOI: 10.3934/fods.2022014. URL: https://www.aimsciences.org/article/id/62e247312d80b75987612297.

Bauer, Ulrich (2015). *Persistent homology algorithm toolbox*. URL: https://github.com/Ripser/ripser.

Carlsson, Gunnar E. et al. (2007). "On the Local Behavior of Spaces of Natural Images". In: *International Journal of Computer Vision* 76, pp. 1–12.

Carriere, Mathieu, Marco Cuturi, and Steve Oudot (2017). "Sliced Wasserstein Kernel for Persistence Diagrams". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 664–673.

Casella, G., R.L. Berger, and Brooks/Cole Publishing Company (2002). *Statistical Inference*. Duxbury advanced series in statistics and decision sciences. Thomson Learning. ISBN: 9780534243128.

Chazal, Frederic and Bertrand Michel (Oct. 2017). "An Introduction to Topological Data Analysis: Fundamental and Practical Aspects for Data Scientists". In: *Frontiers in Artificial Intelligence* 4. DOI: 10.3389/frai.2021.667963.

Cox, Michael A. A. and Trevor F. Cox (2008). "Multidimensional Scaling". In: *Handbook of Data Visualization*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 315–347. ISBN: 978-3-540-33037-0. DOI: 10.1007/978-3-540-33037-0_14. URL: https://doi.org/10.1007/978-3-540-33037-0_14.

Csardi, Gabor and Tamas Nepusz (2006). "The igraph software package for complex network research". In: *InterJournal* Complex Systems, p. 1695. URL: https://igraph.org.

De Silva, Vin, Dmitriy Morozov, and Mikael Vejdemo-Johansson (2011). "Dualities in persistent (co) homology". In: *Inverse Problems* 27.12, p. 124003.

Dhillon, Inderjit S., Yuqiang Guan, and Brian Kulis (2004). "A Unified View of Kernel k-means , Spectral Clustering and Graph Cuts". In.

Eddelbuettel, Dirk and Romain Francois (2011). "Rcpp: Seamless R and C++ Integration". In: *Journal of Statistical Software* 40.8, pp. 1–18. DOI: 10.18637/jss.v040.i08.

Edelsbrunner, Herbert and John Harer (Jan. 2010). *Computational Topology: An Introduction*. American Mathematical Society. ISBN: 978-0-8218-4925-5. DOI: 10.1007/978-3-540-33259-6_7.

Edelsbrunner, Herbert, David Letscher, and Afra Zomorodian (2000). "Topological Persistence and Simplification". In: *Discrete & Computational Geometry* 28, pp. 511–533.

Fasy, Brittany et al. (Mar. 2014). "Confidence Sets for Persistence Diagrams". In: *The Annals of Statistics* 42, pp. 2301–2339.

Fasy, Brittany T. et al. (2021). *TDA: Statistical Tools for Topological Data Analysis*. R package version 1.7.7. URL: https://CRAN.R-project.org/package=TDA.

Gracia-Tabuenca, Zeus et al. (2020). "Topological Data Analysis reveals robust alterations in the whole-brain and frontal lobe functional connectomes in Attention-Deficit/Hyperactivity Disorder". In: *eneuro*.

Gretton, Arthur et al. (2007). "A Kernel Statistical Test of Independence". In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt et al. Vol. 20. Curran Associates, Inc. URL: https : / / proceedings . neurips . cc / paper / 2007 / file / d5cfead94f5350c12c322b5b664544c1-Paper.pdf.

Haim Meirom, Shaked and Omer Bobrowski (May 2022). "Unsupervised Geometric and Topological Approaches for Cross-Lingual Sentence Representation and Comparison". In: *Proceedings of the 7th Workshop on Representation Learning for NLP*. Dublin, Ireland:

Association for Computational Linguistics, pp. 173–183. DOI: 10.18653/v1/2022.repl4nlp-1.18. URL: https://aclanthology.org/2022.repl4nlp-1.18.

Hornik, Kurt (2005). "A CLUE for CLUster Ensembles". In: *Journal of Statistical Software* 14.12. DOI: 10.18637/jss.v014.i12.

Hyekyoung, Lee et al. (Sept. 2014). "Hole detection in metabolic connectivity of Alzheimer's disease using kappa-Laplacian". In: *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*. Vol. 17, pp. 297–304. ISBN: 978-3-319-10442-3. DOI: 10.1007/978-3-319-10443-0_38.

Karatzoglou, Alexandros et al. (2004). "kernlab – An S4 Package for Kernel Methods in R". In: *Journal of Statistical Software* 11.9, pp. 1–20. URL: https://www.jstatsoft.org/v11/i09/.

Kerber, Michael, Dmitriy Morozov, and Arnur Nigmetov (2017). "Geometry Helps to Compare Persistence Diagrams". In: *ACM Journal of Experimental Algorithmics* 22. ISSN: 1084-6654. DOI: 10.1145/3064175. URL: https://doi.org/10.1145/3064175.

Krishnapriyan, Aditi S. et al. (2021). "Machine learning with persistent homology and chemical word embeddings improves prediction accuracy and interpretability in metal-organic frameworks". In: *Scientific Reports* 11.1, p. 8888.

Kusano, Genki, Kenji Fukumizu, and Yasuaki Hiraoka (2018). "Kernel Method for Persistence Diagrams via Kernel Embedding and Weight Factor". In: *Journal of Machine Learning Research* 18.189, pp. 1–41. URL: https://jmlr.org/papers/v18/17-317.html.

Le, Tam and Makoto Yamada (2018). "Persistence Fisher Kernel: A Riemannian Manifold Kernel for Persistence Diagrams". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2018/file/959ab9a0695c467e7caf75431a872e5c-Paper.pdf.

Lee, Hyekyoung et al. (2011). "Discriminative persistent homology of brain networks". In: *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pp. 841–844. DOI: 10.1109/ISBI.2011.5872535.

Murphy, Kevin P (2012). *Machine learning: a probabilistic perspective*. MIT press.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Reininghaus, Jan et al. (2015). "A stable multi-scale kernel for topological machine learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4741–4748.

Robinson, Andrew and Katharine Turner (2017). "Hypothesis testing for topological data analysis". In: *Journal of Applied and Computational Topology* 1 (2).

Scholkopf, Bernhard, Alexander Smola, and Klaus-Robert Muller (1998). "Nonlinear Component Analysis as a Kernel Eigenvalue Problem". In: *Neural Computation* 10, pp. 1299–1319.

Silva, Vin de, Dmitriy Morozov, and Mikael Vejdemo-Johansson (2011). "Persistent Cohomology and Circular Coordinates". In: *Discrete & Computational Geometry* 45.4, pp. 737–759.

Turner, Katharine (July 2013). "Means and medians of sets of persistence diagrams". In: *arXiv*.

Turner, Katharine et al. (2014). "Frechet Means for Distributions of Persistence Diagrams". In: *Discrete & Computational Geometry* 52.1, pp. 44–70.

Ushey, Kevin, JJ Allaire, and Yuan Tang (2022). *reticulate: Interface to 'Python'*. R package version 1.24. URL: https://CRAN.R-project.org/package=reticulate.

Wadhwa, Raoul et al. (2019). *TDAstats: Pipeline for Topological Data Analysis*. R package version 0.4.1. URL: https://github.com/rrrlw/TDAstats.

Yen, Peter Tsung-Wen and Siew Ann Cheong (2021). "Using Topological Data Analysis (TDA) and Persistent Homology to Analyze the Stock Markets in Singapore and Taiwan". In: *Frontiers in Physics* 9. ISSN: 2296-424X. DOI: 10.3389/fphy.2021.572216. URL: https://www.frontiersin.org/article/10.3389/fphy.2021.572216.

Zomorodian, Afra (2010). "The Tidy Set: A Minimal Simplicial Set for Computing Homology of Clique Complexes". In: *Proceedings of the Twenty-Sixth Annual Symposium on Computational Geometry*. SoCG '10. Snowbird, Utah, USA: Association for Computing

Machinery, pp. 257–266. ISBN: 9781450300162. DOI: 10.1145/1810959.1811004. URL: https://doi.org/10.1145/1810959.1811004.

Zomorodian, Afra and Gunnar Carlsson (Feb. 2005). "Computing Persistent Homology". In: *Discrete and Computational Geometry* 33, pp. 249–274. DOI: 10.1007/s00454-004-1146-y.

2.7 Human Connectome Project analysis

2.7.1 Abstract

In the `TDApplied` package vignette "TDApplied theory and practice" simulated data is used to provide examples of package function. In this vignette we will demonstrate that `TDApplied` can carry out meaningful analyses of real (i.e. non-simulated) data which other software packages cannot. We analyzed data from a very well-known neurological dataset, and identified topological features of neurological computation in single and multiple subjects which were correlated with (i) the task the subjects were performing while the data was collected, and (ii) the behavior (i.e. reaction time) of the subjects during the task – these correlations suggest that the features are meaningful in the context of an neuroimaging analysis. Moreover, these features were identified, interpreted and analyzed with the `bootstrap_persistence_diagram`, `vr_graphs` and `diagram_kpca` functions, only the first of which has any implementation in another R package (`TDA`). `TDApplied` is therefore a powerful tool for applied topological analyses of data.

2.7.2 Introduction

A popular technology for studying neural function is called *functional magnetic resonance imaging* (fMRI), in which oxygenated blood-flow across the brain is detected via magnetic resonance over multiple time points; fMRI is a proxy measurement of neural activity. Spatial activity patterns, i.e. vectors of measured values across space in a single time point, are modulated by performing tasks. The *study design* is the sequence of temporal *blocks* of performing these tasks and each task type is called a *condition*, and a study design can evoke meaningful information about neural processing related to tasks. Collections of spatial activity patterns (for example the spatial patterns evoked by a particular task over multiple time points) have previously been analyzed with topological and geometric techniques which capture their global structural features (X. Liu et al., 2013; Saggar et al.,

2018, 2021; Shine et al., 2019). However, these analyses are not designed to capture the spatially periodic features of fMRI data which we would expect to exist in abundance (Caballero-Gaudes and Reynolds, 2017; Greve et al., 2013; T. Liu, 2016). One persistent homology analysis of fMRI spatial pattern data found robust 0-dimensional topological features (i.e. clusters of time points with similar spatial patterns) whose persistence values negatively correlated with fluid intelligence (Anderson et al., 2018). Larger differences between spatial patterns at different time points generally corresponded to lower values of fluid intelligence, but higher-dimensional topological features such as loops were not considered in that analysis.

In this exploratory analysis, using the R package `TDApplied`, we utilized persistent homology to find, in one subject's fMRI data in an emotion task, task-related signal in the form of a spatial loop. We then linked the loop back to the subject's raw data to interpret what neurological features the loop represented. Finally we showed that topological features in 100 subject's emotion task fMRI data were correlated with behavior (i.e. the subject's response time to certain task blocks). While neuroimaging researchers would not consider a single loop within one subject "real" or "significant" without finding a similar loop across multiple subjects, the task of optimally matching loops between datasets is an open problem, so we leave the problem of finding group-level spatial loops to future work. For our analysis we will use data from the famous Human Connectome Project (Glasser, Smith, et al., 2016) which contains extensively preprocessed neuroimaging data from roughly 1200 subjects. We focused on the HCP emotion task data, which alternated between two conditions - deciding which of two faces matched another target face in emotion, and deciding which of two shapes matched another target shape. We only analyzed the right-to-left phase encoding scan (this is a parameter of MRI imaging which determines the ordering of when image slices of the brain are obtained) as this was the phase encoding direction for the specific subject loop we analyzed. Also, all fMRI data was projected onto surface nodes – points on a mesh of the brain's surface geometry – which are more comparable across subjects than standard 3D volumes (Glasser, Smith,

et al., 2016). The script used to perform the analysis can be found in the "exec" directory of `TDApplied`. Our analysis demonstrates the potential of using `TDApplied` for deriving interpretable and otherwise obscured insights from real datasets.

2.7.3 A task-related spatial loop

For HCP subject 103111 we calculated a persistence diagram by analyzing the time-point by time-point correlation matrix $[\rho_{i,j}]$ of spatial patterns. A distance matrix was computed by the transformation $\rho_{i,j} \to \sqrt{2(1-\rho_{i,j})}$ (see the appendix for details), and the bootstrap procedure (Fasy et al., 2014) found a single significant loop. We plotted the VR graph (Zomorodian, 2010) of the loop representative with ϵ being the birth value (with nodes labelled by their time points), and the VR graph of the whole dataset at this ϵ (with nodes in the representative cycle colored red):

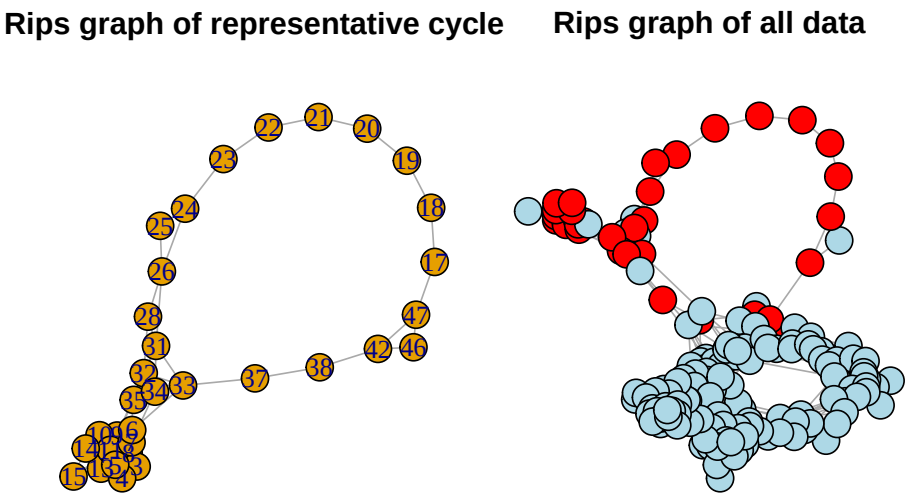

Figure 2.21: The VR graphs of (left) just the representative cycle time points, and (right) all time points, with both epsilon scales at the loop birth value.

Two things are apparent from these plots – the first is that the most persistent loop is the first task block (based on the time points in that block), and the second is that the dataset mainly forms two major loops in a figure eight. The secondary loop was the

second most persistent loop in the diagram, and was comprised of (almost) all other time points. We found that physiology (i.e. breathing, measured in the HCP dataset as the pressure exerted by the subject's abdomen on the sensor belt, and averaged for each graph node) did not account for the two-loop structure, whereas task structure (measured by time-since-last-block – i.e. the length of time since the most recent task block started) did. In these graphs, red represents high values, pink middle-high, white middle, light blue middle-low, blue low and black missing.

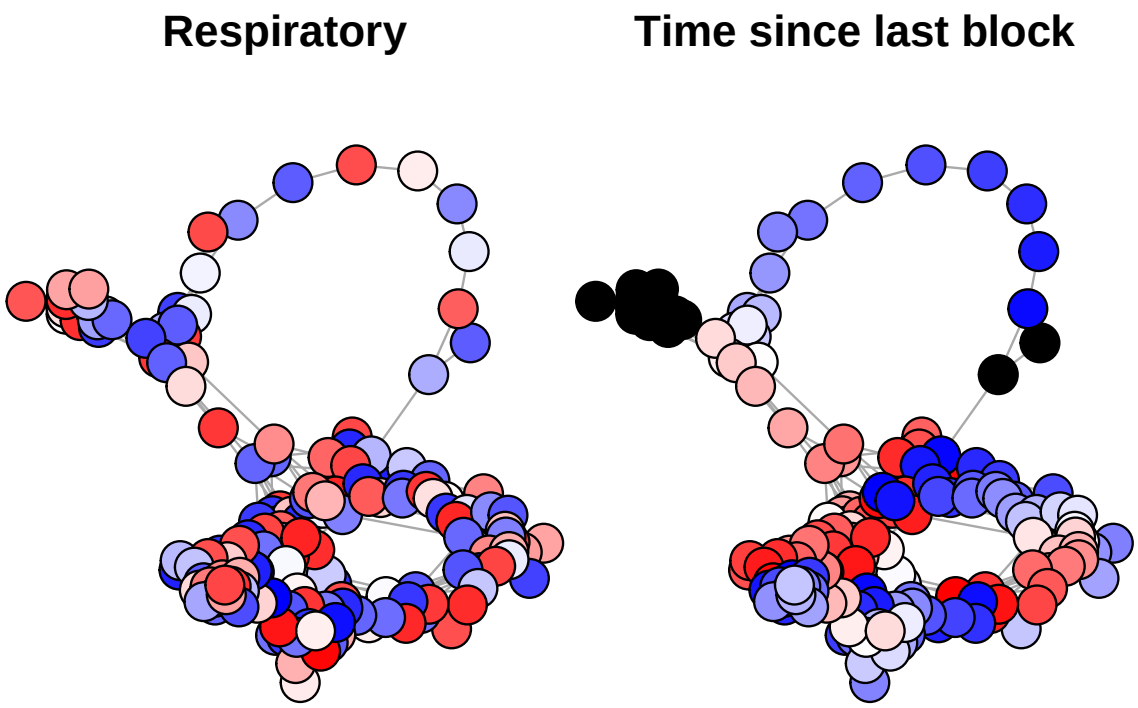

Figure 2.22: VR graph of all time points, colored by (left) mean respiration and (right) time-since-last-block. Only in the right graph do we see clear color clusters or gradients, suggesting that physiology did not account for the structure of the loops whereas task-timing did.

The secondary loop seemed task-related when we colored its VR graph by time-since-last-block (i.e. the length of time since the most recent task block started), with clusters of

similarly-colored time points (i.e. nodes) and smooth gradients at various points around the loop:

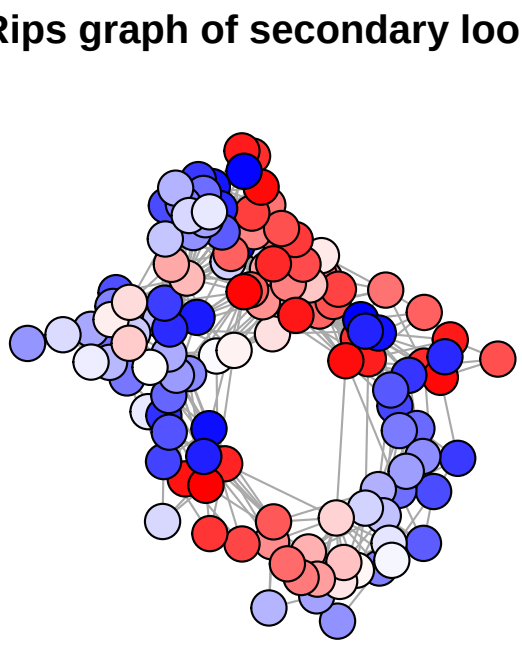

Figure 2.23: VR graph of the secondary loop, colored by time-since-last-block.

2.7.4 Linking the secondary loop to raw data

We next linked the secondary loop back to the fMRI data from which it came. From the 2D layout of its VR graph we computed a θ variable (angle around the loop, between $-\pi$ and π) and an r variable (distance to the origin). Using linear models of the form $A_i = \beta_1 \cos(\theta) + \beta_2 \sin(\theta) + \beta_3$ and $A_i = \beta_4 r + \beta_5$ for the activity A of surface node i, we found that out of the total 91282 surface nodes, the activity of 1475 had a significant relationship with either $\cos(\theta)$ or $\sin(\theta)$, and the activity of 53 had a significant relationship with r (significance thresholding was done at the Bonferroni level of approximately $0.05/(2 * 91282) = 2.74 * 10^{-7}$). These two sets of nodes only shared one common node.

Here are surface plots of the nodes for the left hemisphere, generated using python, responding to θ and r:

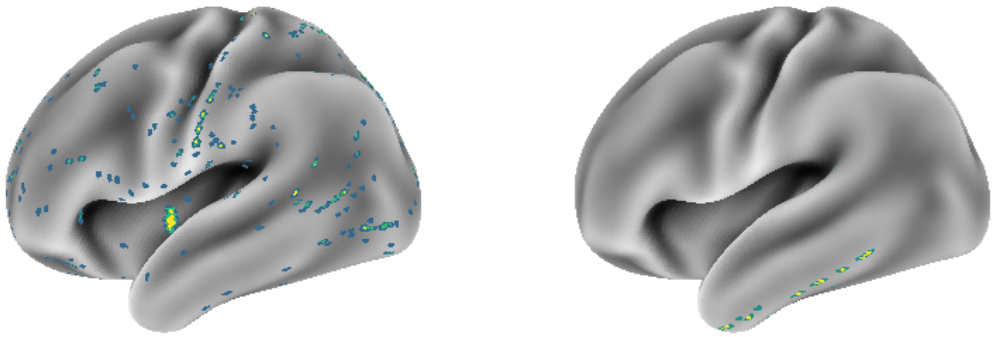

Figure 2.24: Surface nodes whose activity was significantly correlated with (left) theta and (right) r.

A large cluster of surface nodes responding to θ was found in the left hemisphere (and not in the right hemisphere), which appeared to belong to the brain region Pol1 in the insular cortex. Interestingly, Pol1 was not found to show significant differences in activity between the faces and shapes conditions (Glasser, Coalson, et al., 2016), despite the implication of the insular cortex in emotional processing (Gogolla, 2017). This finding suggests that spatial loops of fMRI data may contain complementary information to typical task-based analyses of fMRI data.

A t-test then found a significant difference in mean r values between shape and face blocks:

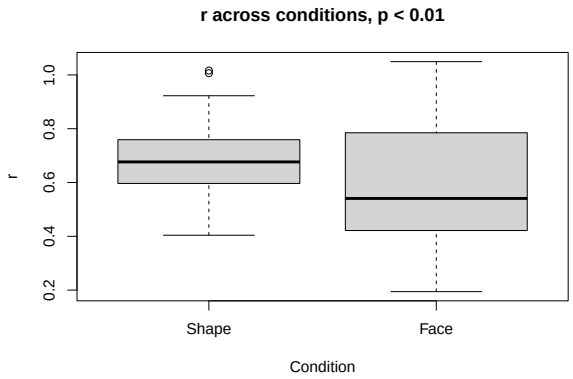

Figure 2.25: Boxplot of r values in shape and face blocks.

2.7.5 Linking topology to behavior

The main goal of using a summary statistic of neurological data, such as persistence diagrams of spatial activity patterns, is to correlate the statistic with behavior. In the HCP dataset, mean reaction time (in milliseconds) was recorded for all subjects and all tasks, so we checked whether topology was predictive of reaction time for the emotion task. We selected 100 subjects at random, computed their emotion right-to-left phase encoding data persistence diagrams, and computed a one dimensional kernel PCA embedding. We then plotted the relationship between the first embedding dimension and reaction time to shape blocks, which had a correlation of about 0.31:

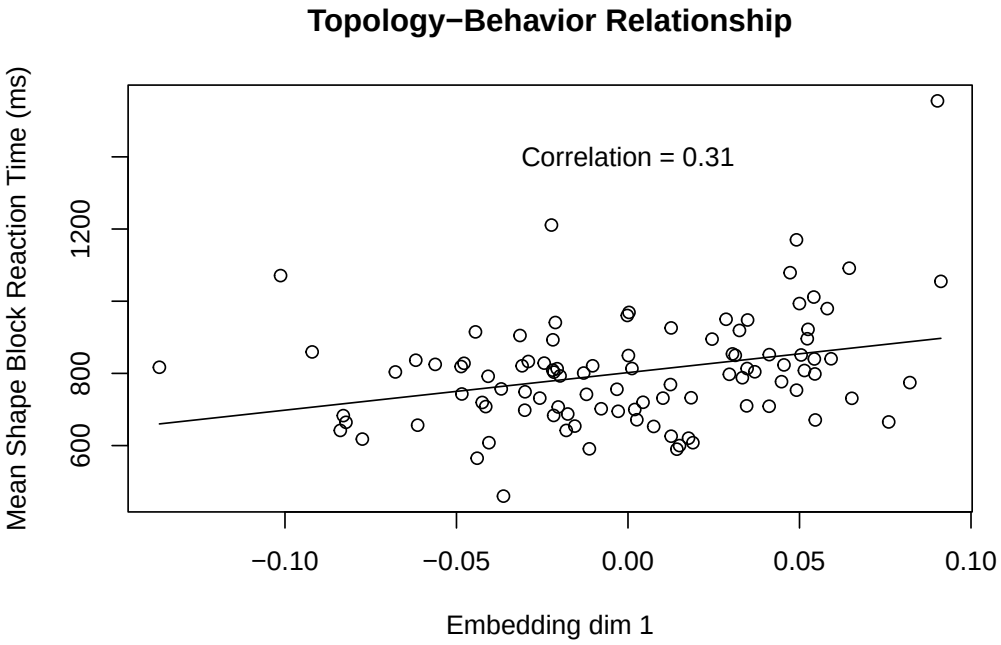

Figure 2.26: A 2D scatterplot, whose x-axis is the 1D PCA embedding coordinates of the 100 subject's emotion persistence diagrams and whose y-axis is the 100 subject's mean response times in the shape blocks trials. The best-fitting regression line is also plotted.

We then computed statistical significance by Fisher-transforming the sample correlation of 0.31 to obtain a test-statistic of about 0.32, and compared this statistic to a normal null distribution with mean 0 and standard error $1/\sqrt{100-3} \approx 0.1$ (Fisher et al., 1936). The resulting p-value, for a two-sided null hypothesis, would have been less than 0.002, suggesting a significant positive correlation between the topology of neural activity spatial patterns and subject behavior.

2.7.6 Conclusion

We analyzed loops of spatial pattern correlations in HCP emotion fMRI data using `TDApplied`, and were able to visualize and interpret significant loops of an individual subject, relating them to the study design. Such a result implies a complex structure for the representation of the conditions, a complexity that cannot be captured by linear models assuming clustered distributions. Moreover, our topology embedding, enabled with this

software and approach, allowed us to discover a relationship between fMRI patterns and reaction time – effectively, capturing a complex brain-behavior relationship. These results point towards the potential usage of 1-dimensional topology in finding task and/or behavior relationships with fMRI or other neuroimaging modalities, again implying that the shape of neuroimaging data may not be completely captured by traditional analysis methods. Moreover, our analysis hinged on the usage of several functions which are (largely) unique to `TDApplied`. These results show that `TDApplied` is the most flexible and useful tool for carrying out meaningful analyses of data.

2.7.7 Appendix: converting correlations to distances

In our analysis of HCP data we converted correlation values to correlation distance values using the formula $\rho \to \sqrt{2 * (1 - \rho)}$. To see why this transformation does produce distance values, let X and Y be two vectors (of the same length n) with 0 mean and unit variance. Then $\rho(X, Y) = \frac{Cov(E,Y)}{\sigma_x \sigma_y} = \frac{E[(X-0)(Y-0)]}{(1)(1)} = E[XY] = \frac{\sum_{i=1}^{n} X_i Y_i}{n}$. The Euclidean distance of X and Y is therefore $d(X, Y) = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2} = \sqrt{\sum_{i=1}^{n} X_i^2 + \sum_{i=1}^{n} Y_i^2 - 2\sum_{i=1}^{n} X_i Y_i} = \sqrt{n + n - 2\rho(X, Y)} = \sqrt{2n(1 - \rho(X, Y))} \propto \sqrt{2(1 - \rho(X, Y))}$. Therefore, since a scaled distance metric (by a positive number like $\sqrt{n}$) is also a distance metric (this is very easy to verify), the transformation $\rho \to \sqrt{2 * (1 - \rho)}$ indeed gives distance values. Note that in Anderson et al., 2018 a proportional transformation was used $\rho \to \sqrt{1 - \rho} \propto \sqrt{2(1 - \rho)}$ and therefore our results are consistent with those found in that work.

2.7.8 References

Anderson, K. L., Anderson, J. S., Palande, S., & Wang, B. (2018). Topological data analysis of functional mri connectivity in time and space domains. In G. Wu, I. Rekik, M. D. Schirmer, A. W. Chung, & B. Munsell (Eds.), *Connectomics in neuroimaging* (pp. 67–77). Springer International Publishing.

Caballero-Gaudes, C., & Reynolds, R. C. (2017). Methods for cleaning the bold fmri signal. *NeuroImage*, *154*, 128–149. https://doi.org/https://doi.org/10.1016/j.neuroimage.2016.12.018

Fasy, B., Lecci, F., Rinaldo, A., Wasserman, L., Balakrishnan, S., & Singh, A. (2014). Confidence sets for persistence diagrams. *The Annals of Statistics*, *42*, 2301–2339.

Fisher, R. A., et al. (1936). Statistical methods for research workers. *Statistical Methods for Research Workers*, (6th Ed).

Glasser, M. F., Coalson, T. S., Robinson, E. C., Hacker, C. D., Harwell, J., Yacoub, E., Ugurbil, K., Andersson, J., Beckmann, C. F., Jenkinson, M., Smith, S. M., & Van Essen, D. C. (2016). A multi-modal parcellation of human cerebral cortex. *Nature*, *536*(7615), 171–178.

Glasser, M. F., Smith, S. M., Marcus, D. S., Andersson, J. L. R., Auerbach, E. J., Behrens, T. E. J., Coalson, T. S., Harms, M. P., Jenkinson, M., Moeller, S., Robinson, E. C., Sotiropoulos, S. N., Xu, J., Yacoub, E., Ugurbil, K., & Van Essen, D. C. (2016). The human connectome project's neuroimaging approach. *Nature Neuroscience*, *19*(9), 1175–1187.

Gogolla, N. (2017). The insular cortex. *Current Biology*, *27*(12), R580–R586. https://doi.org/https://doi.org/10.1016/j.cub.2017.05.010

Greve, D., Brown, G., Mueller, B., Glover, G., Liu, T., & Network, F. B. R. (2013). A survey of the sources of noise in fmri. *Psychometrika*, *78*, 396–416.

Liu, T. (2016). Noise contributions to the fmri signal: An overview. *NeuroImage*, *143*, 141–151.

Liu, X., Chang, C., & Duyn, J. (2013). Decomposition of spontaneous brain activity into distinct fmri co-activation patterns. *Frontiers in Systems Neuroscience*, *7*, 101. https://doi.org/10.3389/fnsys.2013.00101

Saggar, M., Shine, J. M., Liégeois, R., Dosenbach, N. U. F., & Fair, D. (2021). Precision dynamical mapping using topological data analysis reveals a unique hub-like transition state at rest. *bioRxiv*. https://doi.org/10.1101/2021.08.05.455149

Saggar, M., Sporns, O., Gonzalez-Castillo, J., Bandettini, P. A., Carlsson, G. E., Glover, G. H., & Reiss, A. L. (2018). Towards a new approach to reveal dynamical organization of the brain using topological data analysis. *Nature Communications*, *9*.

Shine, J. M., Breakspear, M., Bell, P. T., Ehgoetz Martens, K. A., Shine, R., Koyejo, O., Sporns, O., & Poldrack, R. A. (2019). Human cognition involves the dynamic integration of neural activity and neuromodulatory systems. *Nature Neuroscience*, *22*(2), 289–296.

Zomorodian, A. (2010). The tidy set: A minimal simplicial set for computing homology of clique complexes. *Proceedings of the Twenty-Sixth Annual Symposium on Computational Geometry*, 257–266. https://doi.org/10.1145/1810959.1811004

2.8 Benchmarking and speedups

2.8.1 Introduction

The `TDApplied` package provides a wide variety of tools for performing powerful applied analyses of multiple persistence diagrams, and in order to make these analyses more practical a number of computational speedups have been built-in. These speedups involve other programming languages (Python and C++), parallel computing and intuitive tricks, and result in significant performance gains (compared to other similar R packages). In this vignette we will describe the methods that are used to make `TDApplied` a highly practical and scalable package for applied topological data analysis in R, as well as benchmarking `TDApplied` functions against suitable counterparts. All benchmarking was carried out on a Windows 10 64-bit machine, with an Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz 3.60 GHz processor with 8 cores and 64GB of RAM.

2.8.2 Speedups

2.8.2.1 Parallelization

When performing multiple independent computations, parallelization can increase speed by performing several computations concurrently. `TDApplied` utilizes parallelization in three ways to achieve significant performance gains:

1. Calculating distance and Gram (i.e. kernel) matrices in parallel – these matrices are the backbone, and limiting runtime factors, of all `TDApplied` machine learning and inference methods.
2. Carrying out the bootstrap procedure (to find significant topological features) in parallel – each bootstrap iteration involves an independent distance calculation.
3. Determining the loss-function value in the permutation test procedure, where distances are calculated between each pair of diagrams in the same (permuted) group.

Parallelization is performed in an operating-system agnostic manner, ensuring that these speedups are available to all `TDApplied` users.

2.8.2.2 Fisher information metric approximation

The Fisher information metric (Le and Yamada, 2018) unlocks the door to a number of `TDApplied` machine learning and inference functions via the persistence Fisher kernel. However, the computational complexity of calculating this metric is quadratic in the number of points in its input diagrams (Le and Yamada, 2018), making calculations using diagrams with many points prohibitive. To solve this issue, a fast approximation has been described for the Fisher information metric using the Fast Gauss Transform (Morariu et al., 2008). This approximation reduces the runtime complexity of the distance calculation to be linear – a huge performance gain with large persistence diagrams. The implementation of the Fast Gauss Transform at https://github.com/vmorariu/figtree was copied into `TDApplied` with only slight modifications to interface with `Rcpp` (Eddelbuettel and Francois, 2011), providing the approximation functionality. Note that the "epsilon" parameter in the original C++ code is called `rho` in `TDApplied`'s implementation in order to avoid confusion about error bounds – epsilon usually denotes an error bound (like in the original code). However, `rho` is not itself a bound on the approximation error of the metric itself, but rather for some subcalculations.

To illustrate how significant this speedup can be, we sampled unit spheres and tori (with inner tube radius 0.25 and major radius 0.75) with different numbers of points 100,200,...,1000 (with 10 iterations at each number of points), calculated their persistence diagrams and benchmarked calculating their Fisher information metric in dimensions 0, 1 and 2, with and without approximation. The plot below shows the results (plotting the mean runtime with a 95% confidence interval, which was too small to show for the approximation):

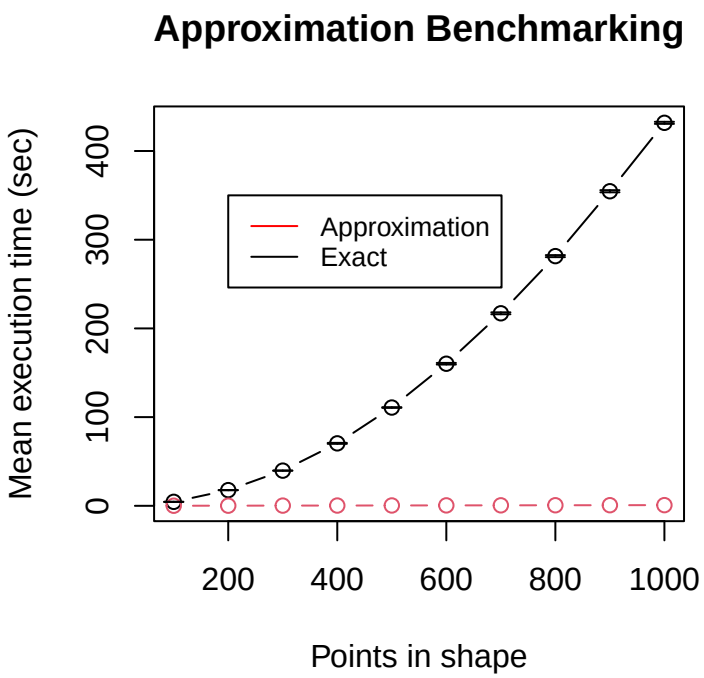

Figure 2.27: A comparison of the mean execution time of comparing persistence diagrams with the exact or approximate Fisher information metric calculation, where diagrams were computed from samples of spheres and tori with varying numbers of points. The approximation was significantly faster, so much so that error bars couldn't be displayed for its plotted points.

A linear model of the runtime ratio (division of the exact vs. approximation mean runtimes) regressed onto the number of points in the shapes found a highly significant positive slope of about 0.57. This means that for every additional 100 points in the shapes the runtime ratio increases by about 0.57 – larger inputs generally lead to greater runtime savings.

Unfortunately this approximation cannot currently be run in parallel in `TDApplied` functions, so in cases where the number of points in the persistence diagrams is small or the number of available cores is large we may consider using the exact calculation instead. However, functions are provided in the "exec/parallel_with_approximation.R" script which can be loaded into your environment to compute distance and Gram matrices

in parallel with approximation (and these matrices can be input into other `TDApplied` functions directly – see the following section).

As a demonstration we will create ten persistence diagrams from circles (100 points points sampled on each) and compute their Fisher information metric distance matrix in parallel with and without approximation:

```
# create 20 diagrams from circles
g <- lapply(X = 1:10,FUN = function(X){

  return(TDAstats::calculate_homology(TDA::circleUnif(100),
                                      dim = 0,threshold = 2))

})

# calculate distance matrices
d_exact <- distance_matrix(g,distance = "fisher",sigma = 1)
d_approx <- parallel_approx_distance_mat(g,rho = 1e-6)
```

The timing difference was impressive – the exact distance matrix was calculated in about 44.8s, whereas the approximate distance matrix was calculated in 1.8s! Moreover, the maximum percent difference between the two matrices was only about 0.9%. When we repeated these calculations with twenty diagrams the timing was 180s for the exact calculation compared to 3.7s with the approximation, and when we used twenty diagrams with each circle having 200 sampled points the timings were 174s and 2.4s for the exact and approximate calculations respectively.

These simulations indicate that the functions `parallel_approx_distance_matrix` and `parallel_approx_gram_matrix` in the "exec" directory can unlock exceptional performance increases in calculating distance/Gram matrices, and can be particularly useful when combined with the speedup documented in the following section. However,

we have found that these parallelized approximation functions can sometimes cause (seemingly non-reproducible, perhaps related to either parallel processing or memory allocation) fatal errors, requiring the user to manually restart the R session. This function should be used carefully, and perhaps not in large loops which may get interrupted. A future update to `TDApplied` should resolve this issue.

2.8.2.3 Using pre-computed matrices

Redundancy can be a huge computational strain when performing multiple related and slow calculations. Applied topological data analysis unfortunately falls victim to this problem since machine learning and inference methods are built on calculating (potentially the same) distance/Gram matrices. In order to circumvent this issue `TDApplied` machine learning and inference functions can accept as input precomputed distance/Gram matrices. Therefore, if a number of analyses are being carried out with the same persistence diagrams and with the same distance/kernel parameter choices (as is often desired) then the distance/Gram matrices can each be computed once and reused across functions.

To illustrate how this works in practice we will use the `generate_TDApplied_vignette_data` function to generate nine persistence diagrams and analyze them with multidimensional scaling (Cox and Cox, 2008), kernel k-means (Dhillon et al., 2004) and kernel principal components analysis (Scholkopf et al., 1998):

```
generate_TDApplied_vignette_data <- function(num_D1,num_D2,num_D3)
{

  # num_D1 is the number of desired copies of D1, and likewise
  # for num_D2 and num_D3

  # create data
```

```
D1 = data.frame(dimension = c(0),birth = c(2),death = c(3))
D2 = data.frame(dimension = c(0),birth = c(2,0),
                death = c(3.3,0.5))
D3 = data.frame(dimension = c(0),birth = c(0),death = c(0.5))

# make noisy copies
noisy_copies <- lapply(X = 1:(num_D1 + num_D2 + num_D3),
                       FUN = function(X){

  # i stores the number of the data frame to make copies of:
  # i = 1 is for D1, i = 2 is for D2 and i = 3 is for D3
  i <- 1
  if(X > num_D1 & X <= num_D1 + num_D2)
  {
    i <- 2
  }
  if(X > num_D1 + num_D2)
  {
    i <- 3
  }
  # store correct data in noisy_copy
  noisy_copy <- get(paste0("D",i))

  # add Gaussian noise to birth and death values
  n <- nrow(noisy_copy)
  noisy_copy$dimension <-
      as.numeric(as.character(noisy_copy$dimension))
  noisy_copy$birth <-
```

```
          noisy_copy$birth + stats::rnorm(n = n,mean = 0,sd = 0.05)
    noisy_copy$death <-
          noisy_copy$death + stats::rnorm(n = n,mean = 0,sd = 0.05)

    # make any birth values which are less than 0 equal 0
    noisy_copy[which(noisy_copy$birth < 0),2] <- 0

    # make any birth values which are greater than their death
    # values equal their death values
    noisy_copy[which(noisy_copy$birth > noisy_copy$death),2] <-
      noisy_copy[which(noisy_copy$birth > noisy_copy$death),3]
    return(noisy_copy)

  })

  # return list containing num_D1 noisy copies of D1, then
  # num_D2 noisy copies of D2, and finally num_D3 noisy copies
  # of D3
  return(noisy_copies)

}

# create noisy copies of D1, D2 and D3
g <- generate_TDApplied_vignette_data(3,3,3)

# calculate MDS embedding
mds <- diagram_mds(diagrams = g,k = 2,dim = 0,sigma = 1.5,
                   distance = "fisher")
```

```
# calculate kmeans clusters with 3 centers
clust <- diagram_kkmeans(diagrams = g,centers = 3,dim = 0,
                         t = 2,sigma = 1.5)

# calculate kpca embedding
pca <- diagram_kpca(diagrams = g,dim = 0,t = 2,sigma = 1.5,
                    features = 2)
```

The time taken to run the MDS, k-means and PCA lines was about 2.53s. Noting that the distance/Gram matrices had shared parameters (i.e. the distance matrix was calculated with the Fisher information metric and the values of t and sigma were all shared), we then repeated the analysis by pre-computing one distance (and Gram) matrix and using these in all three analyses:

```
D <- distance_matrix(diagrams = g,dim = 0,sigma = 1.5,
                     distance = "fisher")
K <- exp(-2*D)
class(K) <- "kernelMatrix"

# calculate MDS embedding
mds <- diagram_mds(D = D,k = 2)

# calculate kmeans clusters with 3 centers
clust <- diagram_kkmeans(diagrams = g,K = K,centers = 3,dim = 0,
                         t = 2,sigma = 1.5)

# calculate kpca embedding
pca <- diagram_kpca(diagrams = g,K = K,dim = 0,t = 2,sigma = 1.5,
```

```
                        features = 2)
```

The new runtime (including calculating the distance and Gram matrices) was about 0.59s, over three times faster than the original. We then repeated the analyses using 300 persistence diagrams, i.e. with

```
# create noisy copies of D1, D2 and D3
g <- generate_TDApplied_vignette_data(100,100,100)
```

The timing scaled proportionally – without using precomputed matrices the time taken was about 121.8s and with precomputed matrices the time taken was about 39.69s. We recommend using precomputed matrices whenever performing multiple analyses of the same persistence diagrams with shared distance/kernel parameters.

2.8.2.4 Storing calculations in `permutation_test`

Another significant source of redundancy can be found in the `permutation_test` function – in each calculation of the loss function, a distance value is computed between each pair of diagrams in the same (possibly permuted) group. However, diagrams will often appear in the same permuted group meaning distances would be needlessly recalculated. To solve this problem the `permutation_test` function creates an initially trivial distance matrix (with entries -1) between all persistence diagrams across all groups, updates its entries when new distance calculations are performed (in parallel as discussed earlier) and retrieves already computed values whenever possible. It is possible to input precomputed distance matrices to this function. However, for standalone usage depending on the group sizes and number of permutations not every pair of diagrams may appear in some permuted group together, so the implemented speedup avoids redundancy without calculating unnecessary distances.

2.8.3 Benchmarking against similar packages

In order to properly situate `TDApplied` in the landscape of software for topological data analysis, we will compare the speed of its calculations to similar calculations from other packages. In the following sections we will benchmark:

(1) Persistent (co)homology calculations with `TDApplied`'s `PyH`, `TDAstats`' (Wadhwa et al., 2019) `calculate_homology` and `rgudhi`'s (Stamm, 2023) `compute_persistence`.
(2) Wasserstein distances between persistence diagrams with `TDApplied`'s `diagram_distance` and `TDA`'s (Fasy et al., 2021) `wasserstein`.
(3) Wasserstein distances between persistence diagrams with `TDApplied`'s `diagram_distance` and the `persim` Python module's (https://persim.scikit-tda.org/en/latest/) `wasserstein`.
(4) Fisher information metric distances between persistence diagrams with `TDApplied`'s `diagram_distance` and `rgudhi`'s `PersistenceFisherDistance`.

The script that was used to perform benchmarking (and plotting the results) is available in the `exec` directory of this package, using `PyH` in certain cases and thus requiring Python. A simple error check is included for the installation of the `reticulate` package, but the script will throw an error if `reticulate` is not properly connected with Python. In order to perform the benchmarking against `rgudhi`, `rgudhi` must be installed explicitly (again requiring configuration with Python) as it is not even a suggested package for `TDApplied` installation. In all cases, benchmarking followed a similar procedure, involving sampling data from simple shapes (unit circles, unit spheres and tori with inner tube radius 0.25 and major radius 0.75) with various number of rows, and performing 10 benchmarking iterations at each number of rows. The mean and standard deviation of run time for the two functions were then calculated at each number of rows.

The benchmarking results are displayed graphically in the following three subsections. On top of comparing raw run time of the various functions, we also compared the scalability of the functions by computing the runtime ratios (i.e. quotient of runtimes in the two packages) of the functions and regressing the ratios onto the number of points in the input shapes. Overall we found that `TDApplied`'s functions are faster and scale better than their R counterparts, and scale similarly to Python counterparts. These results indicate that `TDApplied` is a powerful and efficient tool for applied topological data analysis in R.

2.8.3.1 Benchmarking `PyH` against `TDAstats`' `calculate_homology` function and `rgudhi`'s `compute_persistence` function

The long calculation time of persistence diagrams is likely a large contributing factor to the slow adoption of topological data analysis for applied data science. Much research has been carried out in order to speed up these calculations, but the current state-of-the-art is the persistent cohomology algorithm (De Silva et al., 2011). In R, the `TDAstats`' `calculate_homology` function is the fastest option for persistence diagram calculations (Somasundaram et al., 2021), being a wrapper for the `ripser` (Bauer, 2015) persistent cohomology engine. `TDApplied`'s `PyH` function is a Python wrapper for the same engine, and `rgudhi` also provides a Python cohomology engine via the C++ library `GUDHI` (Lacombe et al., 2019) with its `compute_persistence` function. Note that `rgudhi`'s function required a number of lines of code to use (i.e. to calculate a persistence diagram across all desired dimensions from a dataset) and we benchmarked all of these lines (see the benchmarking script in the "exec" directory for details). We benchmarked all three function's run time on circles, spheres and tori. The results were as follows:

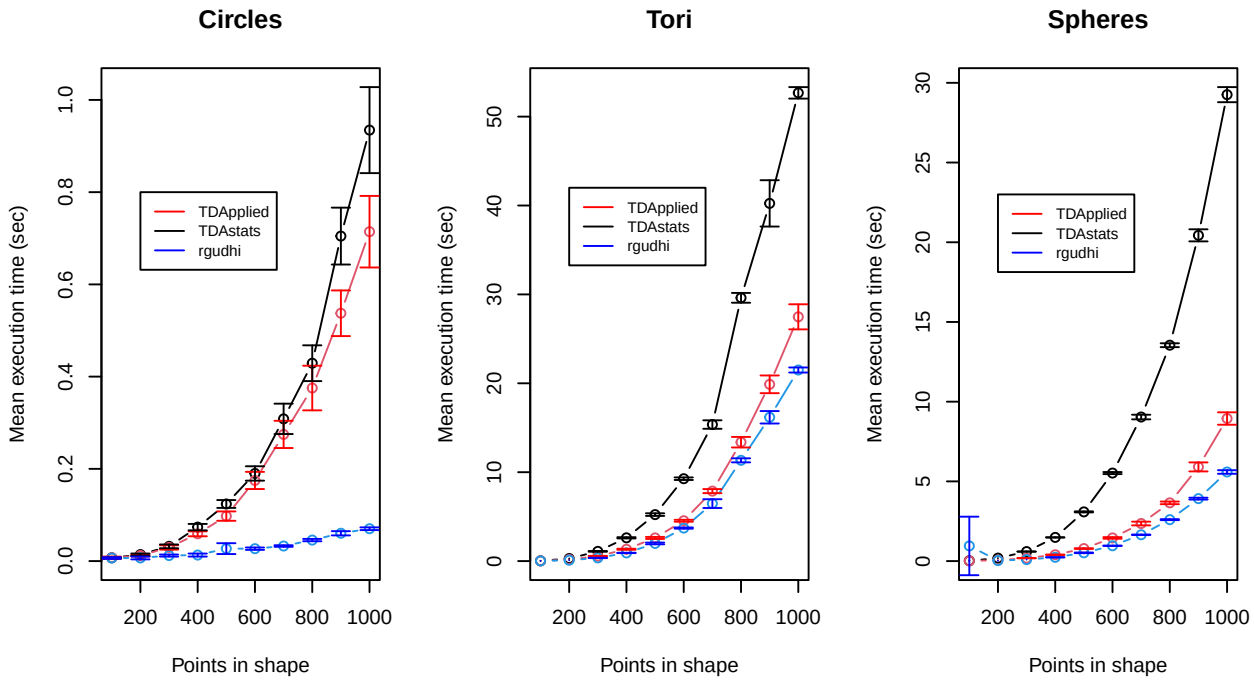

Figure 2.28: Comparisons between the `TDApplied`, `rgudhi` and `TDAstats` homology calculations and simulated datasets of circles (left), tori (middle) and spheres (right). `rgudhi` was the fastest, followed by `TDApplied` and then `TDAstats`. Note the different temporal scalings of the three y-axes - more complex shapes required more compute time for all three packages.

As we can see the run time of `compute_persistence` was fastest, followed by `PyH` and finally `calculate_homology`. We then used linear models to analyze how the three functions scale compared to each other for all three shapes, regressing the ratio of runtimes onto the number of points in the data set. For the models dividing the runtime of `calculate_homology` by that of `PyH`, the intercepts were significant and positive but the slopes were positive and either barely significant or not significant (at the $\alpha = 0.05$ level). This suggests that the runtime of the two functions scale similarly, but there was a constant speed increase of `PyH` which differed by shape (about 1.06 times faster for circles, 1.75 times faster for tori and 3 times faster for spheres). The models for dividing the runtime of `PyH` by that of `compute_persistence` yielded similar results - roughly constant multiplicative runtime decreases with `rgudhi`. Overall, if Python is available to a

TDApplied user then the PyH function may provide speedups compared to the TDAstats function calculate_homology, but in that case rgudhi's compute_persistence is the fastest option at the expense of more lines of code (computing a persistence diagram of just dimension 0 as a data frame takes 5 lines of code).

2.8.3.2 Benchmarking the TDApplied diagram_distance and TDA wasserstein functions

Computing wasserstein (or bottleneck) distances between persistence diagrams is a key feature of some of the main topological data analysis software packages in R and Python. However, these calculations can be very expensive, rendering practical applications of topological data analysis nearly unfeasible. Since TDAstats has implemented an unconventional distance calculation (see the package vignette "Comparing distance calculations" for details), we will benchmark TDApplied's diagram_distance function against the TDA wasserstein function on spheres and tori, calculating their distance in dimensions 0, 1 and 2 and recording the total time. The results were as follows (the 95% confidence interval was too small to be plotted for the diagram_distance function):

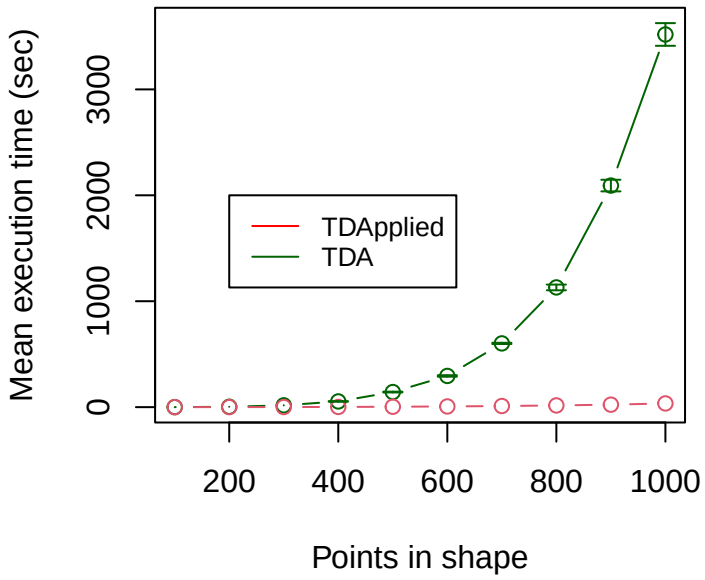

Figure 2.29: A comparison of the mean execution time of `TDApplied` and `TDA` distance functions on persistence diagrams computed from simulated pairs of spheres and tori with varying numbers of data points. `TDApplied` was significantly faster than `TDA` and this difference was so great that no confidence intervals could be seen for `TDApplied`'s plotted points.

A linear model, regressing the ratio of `TDA`'s runtime divided by `TDApplied`'s runtime onto the number of points in the data set, found a significant positive coefficient of the number of points. This suggests that `diagram_distance` scales better than `wasserstein`, and the model estimated a 95x speed up for 1000 data points. These results suggest that distance calculations with `TDApplied` are faster and more scalable, making the applications of statistics and machine learning with persistence diagrams more feasible in R. This is why the `TDA` distance calculation was not used in the `TDApplied` package.

2.8.3.3 Benchmarking the `TDApplied` `diagram_distance` function against `persim`'s `wasserstein` function

While the functionality of Python packages for topological data analysis packages are out of the scope for an R package, in order to fully situate `TDApplied` in the landscape

of topological data analysis software we will benchmark the `diagram_distance` function against its counterpart from the `scikit-TDA` collection of libraries, namely the `wasserstein` function from the `persim` Python module. The R package `reticulate` (Ushey et al., 2022) was used to carry out this benchmarking, via installing, importing and using the `persim` module. This benchmarking procedure also used spheres and tori, calculating distances in dimensions 0, 1 and 2, and the results were as follows (confidence intervals for the `persim` package were too small to be plotted):

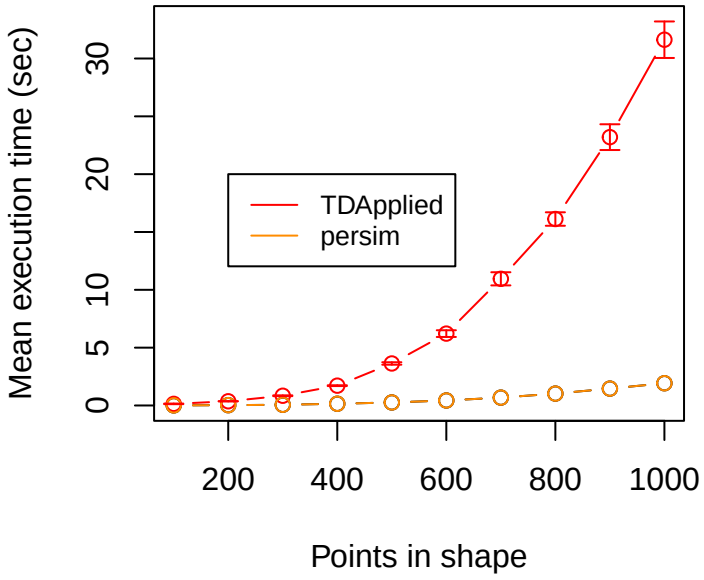

Figure 2.30: A comparison of the mean execution time of `TDApplied` and `persim` distance functions on persistence diagrams computed from simulated pairs of spheres and tori with varying numbers of data points. `persim` was significantly faster than `TDApplied`, to the point that no confidence intervals could be seen for `persim`'s plotted points.

The runtime of the `persim wasserstein` function was significantly faster than `TDApplied`'s `diagram_distance` function. However, a linear model of the runtime ratio of `TDApplied` vs. `persim` against the number of points in the shape finds evidence that the two functions scale similarly, since the estimated coefficient for number of points was not significant but the intercept (15) was highly significant. Nevertheless, the raw

speed increase in Python could be the basis for a very fast Python counterpart to the `TDApplied` package in the future.

2.8.3.4 Benchmarking the `TDApplied` `diagram_distance` and `rgudhi` `PersistenceFisherDistance` functions

Kernel calculations of persistence diagrams open the door for a number of kernel-based machine learning methods, and the fisher information metric is the building block for one such kernel function (Le and Yamada, 2018). Currently, only the `TDApplied` and `rgudhi` R packages provide the functionality for any kernel calculations, and while `rgudhi` provides more kernel functions than `TDApplied`, `rgudhi` requires Python configuration whereas `TDApplied` does not. We will benchmark `TDApplied`'s `diagram_distance` function against `rgudhi`'s `PersistenceFisherDistance` function. We were not able to use the approximation functionality of `rgudhi`'s function (its documentation is missing a reproducible example), so we only benchmarked against `rgudhi`'s exact distance calculation. We again performed benchmarking on spheres and tori, calculating their distance in dimensions 0, 1 and 2 and recording the total time. Only the `TDApplied` approximate distance calculation runtimes (not the exact calculation runtimes) were comparable to those of `rgudhi`'s exact calculation, so here we plot the results of `TDApplied`'s approximate and `rgudhi`'s exact calculations:

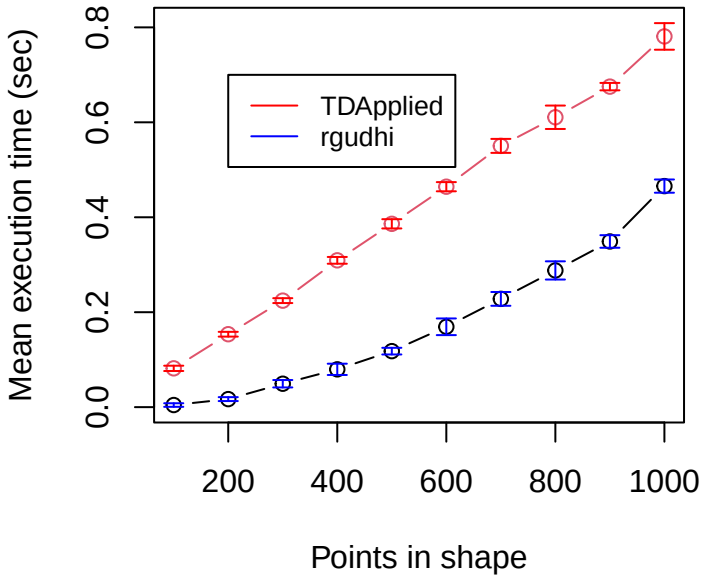

Figure 2.31: A comparison of the mean execution time of `TDApplied` and `rgudhi` distance functions on persistence diagrams computed from simulated pairs of spheres and tori with varying numbers of data points. `rgudhi`'s exact calculations were significantly faster than `TDApplied`'s approximate ones.

The `rgudhi` exact calculations were clearly faster than `TDApplied`'s approximate ones. However, a linear model regressing the ratio of `TDApplied`'s runtime divided by `rgudhi`'s runtime onto the number of points in the data set found a significant negative coefficient of the number of points. This suggests that `diagram_distance` scales better than `PersistenceFisherDistance`, and with enough data points there would be performance gains using the `TDApplied` function. However, if Python configuration for the `rgudhi` package is possible then the `rgudhi` function should be preferred for Fisher information metric calculations for diagrams with not too many points.

An important consequence of these results are that for some analyses we can compute faster distance/Gram matrices with `rgudhi` compared to with `TDApplied`, and these matrices can feed directly into `TDApplied`'s machine learning and inference methods. An example of how to do so can be found in the "Comparing Distance Calculations" package vignette, with functions that can be copied and ran.

2.8.4 Conclusion

`TDApplied` includes a wide variety of functions for machine learning and inference with persistence diagrams; however, these methods can have prohibitively long runtimes. In order to make `TDApplied` functions more practical, a number of speedups have been implemented resulting in substantial performance gains, including parallelization, fast approximation to the Fisher information metric and allowing precomputed distance/Gram matrices to be input to the functions. Benchmarking `TDApplied` functions against suitable counterparts in R situates `TDApplied` as the state-of-the-art in terms of speed for topological data analysis calculations in R. However, comparisons against Python functions (including the `rgudhi` package) indicate that further speedups may be possible. With all its optimizations, `TDApplied` makes applied topological data analysis possible and practical like never before.

2.8.5 References

Bauer, U. (2015). *Persistent homology algorithm toolbox*. https://github.com/Ripser/ripser

Cox, M. A. A., & Cox, T. F. (2008). Multidimensional scaling. In *Handbook of data visualization* (pp. 315–347). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-33037-0_14

De Silva, V., Morozov, D., & Vejdemo-Johansson, M. (2011). Dualities in persistent (co) homology. *Inverse Problems*, *27*(12), 124003.

Dhillon, I. S., Guan, Y., & Kulis, B. (2004). A unified view of kernel k-means , spectral clustering and graph cuts. *UTCS Technical Report*.

Eddelbuettel, D., & Francois, R. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, *40*(8), 1–18. https://doi.org/10.18637/jss.v040.i08

Fasy, B. T., Kim, J., Lecci, F., Maria, C., Millman, D. L., & Rouvreau., V. (2021). *Tda: Statistical tools for topological data analysis* [R package version 1.7.7]. https://CRAN.R-project.org/package=TDA

Lacombe, T., Montassif, H., Soriano-Trigueros, M., Spreeman, G., & Takenouchi, M. (2019). *The gudhi library is a generic open source c++ library, with a python interface, for topological data analysis (tda) and higher dimensional geometry understanding*. https://gudhi.inria.fr/

Le, T., & Yamada, M. (2018). Persistence fisher kernel: A riemannian manifold kernel for persistence diagrams. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/959ab9a0695c467e7caf75431a872e5c-Paper.pdf

Morariu, V., Srinivasan, B., Raykar, V. C., Duraiswami, R., & Davis, L. S. (2008). Automatic online tuning for fast gaussian summation. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in neural information processing systems* (Vol. 21). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2008/file/d96409bf894217686ba124d7356686c9-Paper.pdf

Scholkopf, B., Smola, A., & Muller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

Somasundaram, E. V., Brown, S. E., Litzler, A., Scott, J. G., & Wadhwa, R. R. (2021). The r journal: Benchmarking r packages for calculation of persistent homology [https://doi.org/10.32614/RJ-2021-033]. *The R Journal*, *13*, 184–193. https://doi.org/10.32614/RJ-2021-033

Stamm, A. (2023). *Rgudhi: An interface to the gudhi library for topological data analysis* [R package version 0.2.0]. https://CRAN.R-project.org/package=rgudhi

Ushey, K., Allaire, J., & Tang, Y. (2022). *Reticulate: Interface to 'python'* [R package version 1.24]. https://CRAN.R-project.org/package=reticulate

Wadhwa, R., Dhawan, A., Williamson, D., & Scott, J. (2019). *Tdastats: Pipeline for topological data analysis* [R package version 0.4.1]. https://github.com/rrrlw/TDAstats

2.9 Personalized analyses with `TDApplied`

2.9.1 Introduction

`TDApplied` contains a variety of built-in methods for performing applied analyses of persistence diagrams. However, these methods are by no means a comprehensive list of common tools used in data science. A `TDApplied` user may wish to augment their existing analysis pipelines to include persistence diagrams, which we will call a "personalized analysis." In this vignette we will explore how to use `TDApplied` in personalized analyses using a technique called *vectorization* – assigning vectors to persistence diagrams and using these vectors in downstream analyses. Note that `TDAvec` (Islambekov and Luchinsky, 2022) and `TDAkit` (You and Yu, 2021) are two R packages specifically designed for vectorization analyses, however the methods they implement remove some information in the persistence diagrams (Bubenik, 2015; Hensel et al., 2021) whereas the kernel approach in `TDApplied` does not remove any information. Nevertheless, `TDAvec` and `TDAkit` may also be consulted for performing personalized analyses of persistence diagrams.

In this vignette we will focus on supervised machine learning analyses (i.e. classification and regression tasks), but similar approaches can be used for unsupervised machine learning, inference, etc. Standard supervised learning methods take as input an $n \times f$ feature matrix (each row representing one data point and each column representing a dataset feature) and a label vector. However, suppose that we had a complicated feature matrix which contains persistence diagrams (i.e. *topological features*), numeric and factor features (i.e. *non-topological features*). This data could be incredibly rich in predictive power, but requires special treatment to be used in typical model training pipelines. A pipeline using `TDApplied` would be:

1. First separate the features into topological features $T_1, \ldots, T_k$, each of which is a list of diagrams, and non-topological features which is a $n \times (f - k)$ feature matrix called NT.

2. For each topological feature $T_i = \{D_{i,1}, \ldots, D_{i,n}\}$ we compute its (approximate) $n \times n$ Gram matrix G_i.
3. Column bind all G_i together into a $n \times (kn)$ feature matrix G.
4. Column bind G and NT to get a new $n \times (kn + f - k)$ feature matrix F.
5. We train the model using the feature matrix F and the original labels.

Once we have built our model we may want to make predictions for n' new data points. In order to make predictions we need to convert this new data into a feature matrix resembling F, and this can be done using the same pipeline as above except in step 2 we compute the **cross** Gram matrix (see the package vignette "`TDApplied` theory and practice" for details about Gram and cross Gram matrices) of each T_i' with its corresponding T_i.

The following section provides an example of these two pipelines.

2.9.2 Classification with extreme gradient boosting (XGBoost)

XGBoost (Chen and Guestrin, 2016) is currently one of the most popular and high-performing machine learning models for classification and regression in industry. How could we access this prediction performance from `TDApplied`? Let's start by loading the `xgboost` package (Chen et al., 2023):

```
library(xgboost)
```

The `xgboost` package has an `xgboost` function with a simple interface for training XGBoost models, requiring only a feature matrix and label vector. For our example we will consider the task of predicting which shape each training example came from – a circle, torus or sphere. The features of our data points are two topological features, one persistence diagram sampled from the shape in dimension 1 and the other diagram sampled from dimension 2, two numeric features which are the mean persistence of the two diagrams, and one factor feature which is a random binary vector. We illustrate by creating 30 data points for this example:

```
# create 30 diagrams from circles, tori and spheres
# up to dimension 2
diags <- lapply(X = 1:30,FUN = function(X){

  if(X <= 10)
  {
    return(TDAstats::calculate_homology(TDA::circleUnif(n = 100),
                                        dim = 2,threshold = 2))
  }

  if(X > 10 & X <= 20)
  {
    return(TDAstats::calculate_homology(TDA::torusUnif(n = 100,
                                        a = 0.25,c = 0.75),
                                        dim = 2,threshold = 2))
  }

  if(X > 20)
  {
    return(TDAstats::calculate_homology(TDA::sphereUnif(n = 100,
                                        d = 2),
                                        dim = 2,threshold = 2))
  }

})

# subset into two features, dimension 1 and dimension 2
T1 <- lapply(X = diags,FUN = function(X){
```

```
  df <- X[which(X[,1] == 1),]
  if(!is.matrix(df))
  {
    df <- as.data.frame(t(df))
  }
  return(as.data.frame(df))

})
T2 <- lapply(X = diags,FUN = function(X){

  df <- X[which(X[,1] == 2),]
  if(!is.matrix(df))
  {
    df <- as.data.frame(t(df))
  }
  return(as.data.frame(df))

})

# calculate max persistence of each diagram
max_pers_H1 <- unlist(lapply(X = T1,FUN = function(X){

  return(max(X[[3]] - X[[2]]))

}))
max_pers_H2 <- unlist(lapply(X = T2,FUN = function(X){
```

```
  if(nrow(X) == 0)
  {
    return(0)
  }
  return(max(X[[3]] - X[[2]]))

}))

# create random binary vector
rand_bin <- sample(factor(c("yes","no")),size = 30,replace = T)

# specify data labels, 0 for circle, 1 for torus, 2 for sphere
labs <- rep(c(0,1,2),each = 10)
```

Now that we have constructed our dataset we will follow steps 1 through 5 of the pipeline to train an XGBoost model:

```
# form non-topological feature matrix
NT <- cbind(max_pers_H1,max_pers_H2,rand_bin)

# calculate the approximate Gram matrix for each
# topological feature
G1 <- gram_matrix(diagrams = T1,sigma = 0.01,dim = 1,rho = 0.0001)
G2 <- gram_matrix(diagrams = T2,sigma = 0.01,dim = 2,rho = 0.0001)

# column bind G_i's into 30x60 feature matrix
G <- cbind(G1,G2)

# column bind G and NT into 30x63 feature matrix
```

```
Fmat <- cbind(G,NT)

# fit XGBoost model with maximum 50 boosting iterations
model <- xgboost(data = Fmat,label = rep(c(0,1,2),each = 10),
                 nrounds = 50,verbose = 0,
                 objective = "multi:softmax",num_class = 3)
```

Now that we have fit our model, let's create three new data points:

```
new_diags <-
list(TDAstats::calculate_homology(TDA::circleUnif(n = 100),
                                  dim = 2,threshold = 2),
     TDAstats::calculate_homology(TDA::torusUnif(n = 100,
                                        a = 0.25,c = 0.75),
                                        dim = 2,threshold = 2),
     TDAstats::calculate_homology(TDA::sphereUnif(n = 100,d = 2),
                                        dim = 2,threshold = 2))

# subset into two features, dimension 1 and dimension 2
T1_prime <- lapply(X = new_diags,FUN = function(X){

  df <- X[which(X[,1] == 1),]
  if(!is.matrix(df))
  {
    df <- as.data.frame(t(df))
  }
  return(as.data.frame(df))

})
```

```
T2_prime <- lapply(X = new_diags,FUN = function(X){

  df <- X[which(X[,1] == 2),]
  if(!is.matrix(df))
  {
    df <- as.data.frame(t(df))
  }
  return(as.data.frame(df))

})

# calculate max persistence of each new diagram
max_pers_H1_prime <- unlist(lapply(X = T1_prime,FUN = function(X){

  return(max(X[,3] - X[,2]))

}))
max_pers_H2_prime <- unlist(lapply(X = T2_prime,FUN = function(X){

  if(nrow(X) == 0)
  {
    return(0)
  }
  return(max(X[,3] - X[,2]))

}))

# create random binary vector
```

```
rand_bin_prime <- sample(factor(c("yes","no")),size = 3,
                         replace = T)
```

We can now predict the label of these new data points as follows:

```
# form non-topological feature matrix
NT_prime <- cbind(max_pers_H1_prime,max_pers_H2_prime,
                                    rand_bin_prime)

# calculate the approximate cross Gram matrix for each
# topological feature
G1_prime <- gram_matrix(diagrams = T1_prime,other_diagrams = T1,
                        sigma = 0.01,dim = 1,
                        rho = 0.0001)
G2_prime <- gram_matrix(diagrams = T2_prime,other_diagrams = T2,
                        sigma = 0.01,dim = 2,
                        rho = 0.0001)

# column bind G_i prime's into 3x60 feature matrix
G_prime <- cbind(G1_prime,G2_prime)

# column bind G_prime and NT_prime into 3x63 feature matrix
Fmat_prime <- cbind(G_prime,NT_prime)

# fix column names of Fmat_prime to be the same as Fmat
colnames(Fmat_prime) <- colnames(Fmat)

# predict data labels
stats::predict(model,Fmat_prime)
```

```
## [1] 0 1 2
```

2.9.3 Conclusion

`TDApplied` contains functions which can perform a number of common data analyses with persistence diagrams. However the inability of these functions to analyze multiple features of varying types limits their utility for rich datasets. In this vignette we showed how `TDApplied` can be used to perform flexible supervised learning with topological and non-topological features with XGBoost models, but similar pipelines could be used for unsupervised learning and inference tasks. As such, `TDApplied` can interface with important data science packages to add the value of persistence diagrams to standard data science pipelines.

2.9.4 References

Bubenik, P. (2015). Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, *16*(1), 77–102.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., Li, Y., & Yuan, J. (2023). *Xgboost: Extreme gradient boosting* [R package version 1.7.5.1]. https://CRAN.R-project.org/package=xgboost

Hensel, F., Moor, M., & Rieck, B. (2021). A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, *4*, 681108. https://doi.org/10.3389/frai.2021.681108

Islambekov, U., & Luchinsky, A. (2022). *Tdavec: Vector summaries of persistence diagrams* [R package version 0.1.1]. https://CRAN.R-project.org/package=TDAvec

You, K., & Yu, B. (2021). *Tdakit: Toolkit for topological data analysis* [R package version 0.1.2]. https://CRAN.R-project.org/package=TDAkit

2.10 Comparing distance calculations

2.10.1 Introduction

A number of R packages exist for computing distances between pairs of persistence diagrams, including `TDA` (Fasy et al., 2021), `rgudhi` (Stamm, 2023) and `TDApplied`. Comparing the speed of these calculations was performed in the "Benchmarking and speed" package vignette, but here we treat the more basic question of "are these distance calculations the same across packages?" Through examples we show that the answer is unfortunately no, but through exploration we attempt to reconcile these differences and provide guidelines for using the different packages. Moreover, we include a proof of algorithm correctness for `TDApplied`'s distance function and in the following section we describe why we do not compare `TDApplied`'s distance function against that of the `TDAstats` package (Wadhwa et al., 2019).

2.10.2 `TDAstats`' `phom.dist` function

The `TDAstats` package has a (wasserstein) distance function, `phom.dist`, which is described in its package documentation as being "not meaningful without a null distribution." But what does this mean? We examined the source code for `TDAstats`, i.e. its "R/inference.R file", and found that the internal function `wass_workhorse` on lines 77-103 is the actual distance calculation. In this function, the persistence values (i.e. death - birth) for topological features of two diagrams (and their diagonal projections appended to the opposite diagram) are ordered from largest to smallest. The persistence values are then paired between the two diagrams according to their orderings, and the absolute differences of these ordered persistence values are exponentiated and summed. This algorithm is not guaranteed to produce an optimal matching of the topological features (in the sense of the usual wasserstein cost function), and is missing the final exponentiation (in the case of the 2-wasserstein metric we take the square root of the sum of squared distances). These

differences to the standard wasserstein formulation were clear to the authors of `TDAstats`, and while their `phom.dist` function is not exactly a wasserstein metric it is still a comparison statistic of two persistence diagrams which can be studied with inference techniques. Nevertheless, these differences are the reason why in `TDApplied` documentation we refer to `TDAstats`' distance calculation as being "non-standard" and why comparisons (of calculations and benchmarking) are not made against it.

2.10.3 Examples

In order to compare the distance calculations of `TDA`, `rgudhi` and `TDApplied`, we will specify three simple diagrams (the same D1, D2 and D3 from the package vignette "`TDApplied` theory and practice") and consider distances between each of the three possible pairs. The three diagrams are as follows.

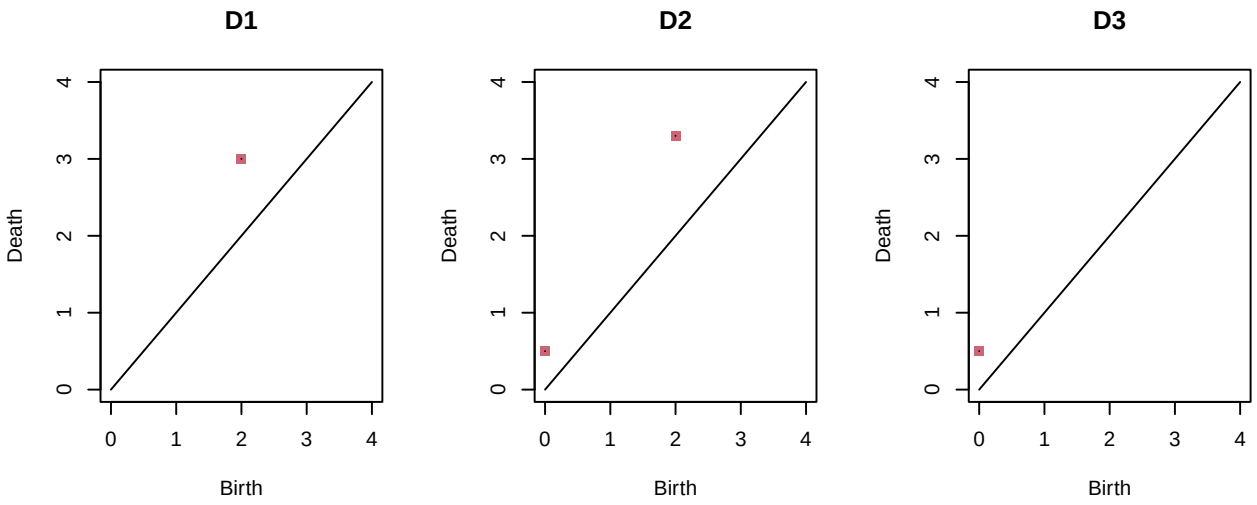

Figure 2.32: Three example persistence diagrams.

D1's point is $(2, 3)$, D2's points are $\{(2, 3.3), (0, 0.5)\}$, and D3's point is $(0, 0.5)$.

In order to compare the distance calculations of the packages we need to have the correct values of the distances. Using the infinity-norm distance for calculating matchings in the

bottleneck and wasserstein distances as in Kerber et al., 2017 and Edelsbrunner and Harer, 2010, we get the following optimal matchings and distance values:

1. Between D1 and D2 we match D1's $(2, 3)$ with D2's $(2, 3.3)$, and D2's $(0, 0.5)$ with its diagonal projection $(0.25, 0.25)$. Therefore, the bottleneck distance is 0.3 and the wasserstein distance is $\sqrt{0.3^2 + 0.25^2} = \sqrt{0.1525} \approx 0.3905125$.
2. Between D1 and D3 we match D1's $(2, 3)$ with its diagonal projection $(2.5, 2.5)$, and D3's $(0, 0.5)$ with its diagonal projection $(0.25, 0.25)$. Therefore, the bottleneck distance is 0.5 and the wasserstein distance is $\sqrt{0.5^2 + 0.25^2} = \sqrt{0.3125} \approx 0.559017$.
3. Between D2 and D3 we match D2's $(0, 0.5)$ with D3's $(0, 0.5)$, and D2's $(2, 3.3)$ with its diagonal projection $(2.65, 2.65)$. Therefore, the bottleneck distance is 0.65 and the wasserstein distance is $\sqrt{0.65^2} = 0.65$.

For the Fisher information metric (Le and Yamada, 2018) we will use the parameter $\sigma = 1$ for simplicity. We must first calculate vectors ρ_1 and ρ_2, normalize them by dividing by the sum of their respective elements, then compute $\cos^{-1}(\sqrt{\rho_1} \cdot \sqrt{\rho_2})$ (where $\cos^{-1}$ is the arccos function). See Le and Yamada, 2018 for computational details.

1. Between D1 and D2 we augment D1 to contain D2's projection points and vice versa, obtaining diagrams $D_1' = \{(2, 3), (2.65, 2.65), (0.25, 0.25)\}$ and $D_2' = \{(2, 3.3), (0, 0.5), (2.5, 2.5)\}$. We compute ρ_1 and ρ_2 as

$$
\begin{aligned}
\rho_1 = \{&\exp(0) + \exp(-0.545/2) + \exp(-10.625/2),\\
&\exp(-0.545/2) + \exp(0) + \exp(-11.52/2),\\
&\exp(-10.625/2) + \exp(-11.52/2) + \exp(0),\\
&\exp(-0.09/2) + \exp(-0.845/2) + \exp(-12.365/2),\\
&\exp(-10.25/2) + \exp(-11.645/2) + \exp(-0.125/2),\\
&\exp(-0.5/2) + \exp(-0.045/2) + \exp(-10.125/2)\}/(2\pi)
\end{aligned}
$$

$$\begin{aligned}\rho_2 =&\{\exp(-0.09/2) + \exp(-10.25/2) + \exp(-0.5/2),\\ &\exp(-0.845/2) + \exp(-11.645/2) + \exp(-0.045/2),\\ &\exp(-12.365/2) + \exp(-0.125/2) + \exp(-10.125/2),\\ &\exp(0) + \exp(-11.84/2) + \exp(-0.89/2),\\ &\exp(-11.84/2) + \exp(0) + \exp(-10.25/2),\\ &\exp(-0.89/2) + \exp(-10.25/2) + \exp(0)\}/(2\pi)\end{aligned}$$

Therefore the arccos of the dot product of the square root of the sum-normalized vectors is approximately 0.02354624.

2. Between D1 and D3 we augment D1 to contain D3's projection point and vice versa, obtaining diagrams $D_1' = \{(2,3),(0.25,0.25)\}$ and $D_3' = \{(0,0.5),(2.5,2.5)\}$. We compute ρ_1 and ρ_2 as

$$\begin{aligned}\rho_1 =&\{\exp(0) + \exp(-10.625/2),\\ &\exp(-10.625/2) + \exp(0),\\ &\exp(-10.25/2) + \exp(-0.125/2),\\ &\exp(-0.5/2) + \exp(-10.125/2)\}/(2\pi)\end{aligned}$$

$$\begin{aligned}\rho_2 =&\{\exp(-10.25/2) + \exp(-0.5/2),\\ &\exp(-0.125/2) + \exp(-10.125/2),\\ &\exp(0) + \exp(-10.25/2),\\ &\exp(-10.25/2) + \exp(0)\}/(2\pi)\end{aligned}$$

Therefore the arccos of the dot product of the square root of the sum-normalized vectors is approximately 0.08821907.

3. Between D2 and D3 we augment D2 to contain D3's projection point and vice versa, obtaining diagrams $D_2' = \{(2, 3.3), (0, 0.5), (0.25, 0.25)\}$ and $D_3' = \{(0, 0.5), (2.65, 2.65), (0.25, 0.25)\}$. We compute ρ_1 and ρ_2 as

$$\begin{aligned}\rho_1 =&\{\exp(0) + \exp(-11.84/2) + \exp(-12.365/2),\\ &\exp(-11.84/2) + \exp(0) + \exp(-0.125/2),\\ &\exp(-12.365/2) + \exp(-0.125/2) + \exp(0),\\ &\exp(-11.84/2) + \exp(0) + \exp(-0.125/2),\\ &\exp(-0.845/2) + \exp(-11.645/2) + \exp(-11.52/2),\\ &\exp(-12.365/2) + \exp(-0.125/2) + \exp(0)\}/\sqrt{2\pi}\end{aligned}$$

$$\begin{aligned}\rho_2 =&\{\exp(-11.84/2) + \exp(-0.845/2) + \exp(-12.365/2),\\ &\exp(0) + \exp(-11.645/2) + \exp(-0.125/2),\\ &\exp(-0.125/2) + \exp(-11.52/2) + \exp(0),\\ &\exp(0) + \exp(-11.645/2) + \exp(-0.125/2),\\ &\exp(-11.645/2) + \exp(0) + \exp(-11.52/2),\\ &\exp(-0.125/2) + \exp(-11.52/2) + \exp(0)\}/\sqrt{2\pi}\end{aligned}$$

Therefore the arccos of the dot product of the square root of the sum-normalized vectors is approximately 0.08741134.

2.10.4 Comparisons

While all three of `TDA`, `rgudhi` and `TDApplied` provide functions for the bottleneck and wasserstein calculations, only `TDApplied` and `rgudhi` have the functionality to calculate the Fisher information metric. We calculated the results of each distance calculation, for each pair of D_i and D_j ($i \neq j$) and each package, and stored the results in tables according to the three distance metrics under comparison:

Bottleneck comparison:

```
##         pair ground_truth TDApplied  TDA rgudhi
## 1 D1 and D2         0.30      0.30 0.30   0.30
## 2 D1 and D3         0.50      0.50 0.50   0.50
## 3 D2 and D3         0.65      0.65 0.65   0.65
```

Wasserstein comparison:

```
##        pair ground_truth TDApplied  TDA rgudhi
## 1 D1 and D2    0.3905125 0.3905125 0.55   0.55
## 2 D1 and D3    0.5590170 0.5590170 0.75   0.75
## 3 D2 and D3    0.6500000 0.6500000 0.65   0.65
```

Fisher information metric comparison:

```
##        pair ground_truth  TDApplied     rgudhi
## 1 D1 and D2   0.02354624 0.02354624 0.02354624
## 2 D1 and D3   0.08821907 0.08821907 0.08821907
## 3 D2 and D3   0.08741134 0.08741134 0.08741134
```

All three packages agreed on the value of the three bottleneck calculations, and both `TDApplied` and `rgudhi` agreed on all Fisher information metric calculations. However, while `TDA` and `TDAstats` agreed on the three wasserstein calculations, two of these differed from `TDApplied`'s output. This occurred because `TDA` and `rgudhi` use a slightly different formula for computing wasserstein distances – where the distance between matched pairs of persistence diagram points is Euclidean rather than an infinity-norm distance. This is a perfectly suitable distance metric and matches the formula in Robinson and Turner, 2017. However, it is different from the published formulas in important works like Kerber et al., 2017 and Edelsbrunner and Harer, 2010 (which are the formulas that `TDApplied` implements).

We state a quick last note of comparison between the kernel calculations in `TDApplied` and `rgudhi`. Even though their Fisher information metric calculations appear to be the same (perhaps up to small differences in algorithm precision), it turns out that `rgudhi` and `TDApplied` return drastically different kernel values. For example, the Fisher information metric between D2 and D3 was (correctly) stated as 0.08741134 for both packages. It follows that when $t = 2$ the persistence Fisher kernel value should be

$\exp(-2 * 0.08741134) \approx 0.8396059$, which is the exact value of the `TDApplied` calculation `diagram_kernel(D2,D3,dim = 0,t = 2)`. However, the code

```
gudhi_kern <- rgudhi::PersistenceFisherKernel$new(bandwidth = 0.5)
gudhi_kern$apply(D2[,2:3],D3[,2:3])
```

returns the value 0.7550683 (note that the `bandwidth` parameter is $1/t$). Even more perplexing is that the following code

```
gudhi_kern <-
       rgudhi::PersistenceFisherKernel$new(bandwidth_fisher = 0.5)
gudhi_kern$apply(D2[,2:3],D3[,2:3])
```

returns the value 1.8326, which should not be possible for the function $\exp(-t * d_{FIM})$ as t and d_{FIM} are always positive and non-negative respectively (so the maximum value should be 1). Unfortunately we were not able to identify the source of this confusion by examining the source code of `rgudhi` (and `GUDHI`), but it is still possible to calculate correct kernel values as follows:

```
d <- rgudhi::PersistenceFisherDistance$new() # sigma = 1
t <- 2 # or whatever desired parameter
exp(-t*d$apply(D2[,2:3],D3[,2:3]))
```

Since distance calculations are much faster with `rgudhi` than with `TDApplied` (see the package vignette "Benchmarking and speedups"), and because distance and Gram matrices can be precomputed and reused across multiple analyses (again see "Benchmarking and speedups") it would be desirable to have a (correct) `rgudhi` Gram matrix function. An example of such a function is the following:

```
# create rgudhi distance object
# sigma = 0.01
gudhi_dist <-
```

```
rgudhi::PersistenceFisherDistance$new(bandwidth = 0.01,
                                      n_jobs = 1L)

# create list of diagrams, only birth and death values in
# dimension 1
g <- lapply(X = 1:10,FUN = function(X){

 df <- diagram_to_df(TDA::ripsDiag(X = TDA::circleUnif(n = 50),
                                   maxdimension = 1,maxscale = 2))
 return(df[which(df[,1] == 1),2:3])

})

# distance matrix function
# diagrams is a list of diagrams (only birth and death columns)
# gudhi_dist is the rgudhi distance object
gudhi_distance_matrix <- function(diagrams,gudhi_dist){

  # get number of rows of each diagram since rgudhi can't
  # calculate distances with empty diagrams
  rows <- unlist(lapply(diagrams,FUN = nrow))
  inds <- which(rows > 0)

  # if inds is empty then return 0 matrix
  if(length(inds) == 0)
  {
    return(matrix(data = 0,nrow = length(diagrams),
                  ncol = length(diagrams)))
```

```
}

# calculate distance matrix for non-zero-row diagrams
d_non_zero <- gudhi_dist$fit_transform(diagrams[inds])

# fix diagonal which can sometimes have non-zero entries
diag(d_non_zero) <- rep(0,nrow(d_non_zero))

# symmetrize (necessary due to numeric rounding issues)
d_non_zero[which(upper.tri(d_non_zero),arr.ind = T)
           [,c("col","row")]] <-
                   d_non_zero[upper.tri(d_non_zero)]

# if all diagrams had at least one row, return
if(length(inds) == length(diagrams))
{
  return(d_non_zero)
}

# create empty distance matrix d
d <- matrix(data = 0,nrow = length(diagrams),
              ncol = length(diagrams))

# update entries of d
e <- as.matrix(expand.grid(inds,inds))
e <- e[which(e[,1] < e[,2]),]
if(!is.matrix(e))
{
```

```
    e <- t(as.matrix(e))
  }
  d[e] <- d_non_zero[which(upper.tri(d_non_zero),arr.ind = T)]
  e <- e[,2:1]
  if(!is.matrix(e))
  {
    e <- t(as.matrix(e))
  }
  d[e] <- d_non_zero[which(upper.tri(d_non_zero),arr.ind = T)]

  return(d)

}

# Gram matrix function
# diagrams is a list of diagrams (only birth and death columns)
# t is the t parameter like in diagram_kernel
# gudhi_dist is the rgudhi distance object
gudhi_gram_matrix <- function(diagrams,t,gudhi_dist){

  # calculate distance matrix
  D <- gudhi_distance_matrix(diagrams = diagrams,
                             gudhi_dist = gudhi_dist)
  return(exp(-t*D))

}

# calculate the Gram matrix
```

```
G <- gudhi_gram_matrix(diagrams = g,t = 1,gudhi_dist = gudhi_dist)
```

Another issue with `rgudhi` calculations can be reproduced as follows:

```
# create data frame
D = data.frame(dimension = c(0,0),birth = c(0,0),
               death = c(1.089866,1.098640))

# create rgudhi distance object
gudhi_dist <-
rgudhi::PersistenceFisherDistance$new(bandwidth = 0.01,n_jobs = 1)

# compute distance
gudhi_dist$apply(D[,2:3],D[,2:3])
```

```
## [1] 0.00000001490116
```

This calculation returns a non-zero number for the distance value of D with itself. This is likely due to numerical rounding issues, and its discovery in `rgudhi` has led to an update in `TDApplied`'s `diagram_distance` function which now returns 0 for this calculation:

```
diagram_distance(D,D,distance = "fisher",sigma = 0.001)
```

```
## [1] 0
```

2.10.5 Proof of correctness for `TDApplied`'s `diagram_distance` function

Even though the Hungarian algorithm can be used to solve the linear sum assignment problem (LSAP) (Hornik, 2005), finding a minimal cost matching of two sets of points, some work needs to be done to properly apply the algorithm to calculate wasserstein or bottleneck distances. For an example we will consider the bottleneck distance, although

the argument still holds with a simple change for the wasserstein distance (squaring matrix entries). Let Diag1 and Diag2 be two diagrams, with n_1 and n_2 points respectively, whose projections onto the diagonal are denoted by $\pi(\text{Diag1})$ and $\pi(\text{Diag2})$ respectively. Then take M to be the following $(n_1 + n_2) \times (n_1 + n_2)$ matrix:

$$M = \left[\begin{array}{c|c} d_\infty(\text{Diag1}, \text{Diag2}) & d_\infty(\text{Diag1}, \pi(\text{Diag2})) \\ \hline d_\infty(\pi(\text{Diag1}), \text{Diag2}) & 0 \end{array}\right]$$

Each row corresponds to the n_1 points in Diag1 followed by the n_2 projections $\pi(\text{Diag2})$, and vice versa for the columns. Then we claim that the solution of the LSAP problem on M has the same cost as the bottleneck distance value between Diag1 and Diag2.

Firstly, we claim that a solution to the LSAP problem on M has a cost which is no less than the distance value. Let the distance value be s. Now suppose, to reach a contradiction, that there existed a lower-cost matching for the LSAP problem for M, m,of cost $s' < s$. Since projection points are matched together with cost 0 in M, let m' contain all the matches in m which are not between two projection points. Then m' would be matching for the distance calculation which has lower cost than s, contradicting the minimality of s. Therefore, a solution to the LSAP problem on M has a cost which is no less than the distance value.

Next, we claim that a solution to the LSAP problem on M has a cost which is no greater than the distance value. Now suppose, to reach a contradiction, that the solution to the LSAP on problem M had cost s, which was larger than the real distance value, s'. Let m' be a matching for the distance calculation of s'. Then since each point in either diagram is either paired with a point in the other diagram or its own diagonal projection, there must be an equal number of unpaired points in both diagrams in m'. Therefore, we can augment m' to a matching m on M in which the unpaired diagonal points are arbitrarily paired up with cost 0. Thus, m has cost $s' < s$, contradicting the minimality of s. Therefore, a solution to the LSAP problem on M has a cost which is no greater than the distance value.

Therefore, a solution to the LSAP problem for M has a cost which is both greater than and less than the bottleneck distance value, and hence the two values must be equal.

2.10.6 References

Edelsbrunner, H., & Harer, J. (2010, January). *Computational topology: An introduction*. American Mathematical Society. https://doi.org/10.1007/978-3-540-33259-6_7

Fasy, B. T., Kim, J., Lecci, F., Maria, C., Millman, D. L., & Rouvreau., V. (2021). *Tda: Statistical tools for topological data analysis* [R package version 1.7.7]. https://CRAN.R-project.org/package=TDA

Hornik, K. (2005). A CLUE for CLUster Ensembles. *Journal of Statistical Software*, *14*(12). https://doi.org/10.18637/jss.v014.i12

Kerber, M., Morozov, D., & Nigmetov, A. (2017). Geometry helps to compare persistence diagrams. *ACM Journal of Experimental Algorithmics*, *22*. https://doi.org/10.1145/3064175

Le, T., & Yamada, M. (2018). Persistence fisher kernel: A riemannian manifold kernel for persistence diagrams. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/959ab9a0695c467e7caf75431a872e5c-Paper.pdf

Robinson, A., & Turner, K. (2017). Hypothesis testing for topological data analysis. *Journal of Applied and Computational Topology*, *1*.

Stamm, A. (2023). *Rgudhi: An interface to the gudhi library for topological data analysis* [R package version 0.2.0]. https://CRAN.R-project.org/package=rgudhi

Wadhwa, R., Dhawan, A., Williamson, D., & Scott, J. (2019). *Tdastats: Pipeline for topological data analysis* [R package version 0.4.1]. https://github.com/rrrlw/TDAstats

Applied topology of representations

Chapter 2 demonstrated how `TDApplied` can be used to carry out various RSA-like analyses on persistence diagrams, but with the additional interpretive abilities of certain persistent-homology-specific tools (like VR graphs) researchers can dive deeper into what topological features of representational spaces mean. However, to this point I have not formally defined RTA as a framework for neuroimaging analyses nor have I shown experimentally that RTA provides insights which RSA misses. There are also special considerations for RSA-type experiments that require attention and discussion which are outside the scope of computational topology, for example how to interpret the topology of RDMs. In Chapter 3 I will define the RTA framework, demonstrate its unique value compared to RSA in two studies, and discuss practical issues related to its application and interpretation in RSA-like studies.

Chapter 3

Representational topology analysis

3.1 Preamble

In Chapter 2 I implemented the necessary computational tools to make persistence diagrams fit into common analysis pipelines of RDMs in RSA. In this chapter I will introduce the representational topology analysis (RTA) framework for capturing and comparing the shape of representational geometry and use the software `TDApplied` in applications. In this new framework topological similarities and differences which are missed by RSA can be identified, and comparisons can be linked to topological features which live "within" the RDMs which generated them. This approach can therefore provide more interpretable and trustworthy comparisons of representational geometries, while avoiding some of the major pitfalls of RSA addressed in Chapter 1 (like how similarity of RDMs does not imply computational similarity). I first provide an example of two representational spaces with similar geometries but different topologies, and explain how judging these spaces to be similar would be biologically incorrect. I then demonstrate the utility of RTA on two studies – a reanalysis of a famous RSA vision study and a naturalistic movie viewing fMRI study – by identifying topological features of neural computation that differentiated between species/regions and which were undetected by RSA. By introducing a novel data visualization technique called the *Proximity Labelled Rips Graph* the topological features

identified by RTA can be visualized and linked with stimulus features, effectively allowing for the segmentation of representational spaces into neural features of computation. This paper has been submitted to the proceedings of the national academy of science (PNAS) journal.

Submitted to PNAS, 2024.

The Topology of representational geometry

Shael Brown[1] and Reza Farivar[2]

[1]Department of Quantitative Life Sciences, McGill University, Montreal Canada.

[2]McGill Vision Research, Department of Opthamology, McGill University, Montreal Canada.

Keywords: Representational similarity analysis — Topological data analysis — Persistent homology — Representational geometry — Object representation — Human — Macaque — fMRI

Acknowledgements: We would like to thank Prof. Nikolaus Kriegeskorte for his invaluable feedback on this research. S.B. and R.F. acknowledge funding from the CIHR 2016 grant for cortical mechanisms of 3-D scene and object recognition in the primate brain.

3.2 Abstract

Representational similarity analysis (RSA) is a powerful tool for abstracting and then comparing neural representations across brains, regions, models and modalities. However, typical RSA analyses compares pairs of representational dissimilarities to judge similarity of two neural systems, and we argue that such methods can not capture the shape of representational spaces. By leveraging tools from computational topology, which can probe the shape of n-dimensional data, we augment RSA to be able to detect more subtle yet real differences and similarities of representational geometries. This new method could be used in conjunction with regular RSA in order to make new inferences about neural function.

3.3 Significance statement

Big data in high-dimensional spaces, like neuroimaging datasets, contain important shape structures. These shape structures can be analyzed to identify the underlying features and dynamics which drive the system. We showed that such analyses, applied to neural activity patterns elicited by viewing various objects, can identify real but subtle and complex features of those objects which are encoded in the brain.

3.4 Introduction

Comparisons of object representations in human cortex can give meaningful insight into the neural mechanisms which encode them, and *representational similarity analysis* (RSA) is a popular framework for organizing and analyzing many such comparisons. RSA estimates the *representational geometry* of a (neural) computational system as a matrix of representational similarities (RSM) or dissimilarities (RDM) (Kriegeskorte and Diedrichsen, 2019). Two such matrices can then be compared with a *second order isomorphism* to quantify similarity or differences between the two systems – Spearman correlation is com-

monly used in neuroimaging studies (Kriegeskorte, Mur, and Bandettini, 2008; Shepard and Chipman, 1970). Correlation between RDMs can identify (i) brain regions which similarly represent stimuli, (ii) commonalities in neural codes between species, (iii) computational models which faithfully represent the function of a brain region, and more.

The powerful and flexible machinery of RSA has yielded many successes in neurosciences and neuroimaging in particular – the introductory paper Kriegeskorte, Mur, and Bandettini, 2008 has been cited over 3300 times so far. More pertinent to the current discussion are examples of RSA applied to fMRI vision studies – one study, Connolly et al., 2012, showed that there may be a spectrum of representations of animals in human visual cortex, from most animate to least; another study, Bracci and Op de Beeck, 2016, showed that visual areas represent shape and category to different extents and with interactions; and in Kriegeskorte, Mur, Ruff, et al., 2008 it was shown that primates may have a similar neural code for object representations in the IT cortex.

However, there have been several major criticisms of RSA, the strongest being that two computational systems with highly similar RDMs may be carrying out their computations in fundamentally different ways (X. Chen et al., 2021; Dujmović et al., 2022). With the primary goal of RSA being to infer whether two computational processes are similar or not, this issue alone may render our inferences from RSA experiments as limited. For example, in Figure 3.1 we consider a representational space of a torus, i.e. a hollow doughnut. Projected into two dimensions (for instance using multidimensional scaling) the torus becomes a 2D annulus (i.e. a circle with added noise) and representations sampled from the torus project to representations in the annulus. Yet the RDMs of the two sets of representations, one set on the surface of the torus (in 3D) and the other along the 2D annulus are erroneously equated by RSA. This example demonstrates that the central assumption of RSA – that linear distances of matrices sufficiently captures similarity of representational geometry for comparison – is not necessarily true.

While our example may seem artificial, orientation-selective neurons in V1 have the representational space of a loop (Singh et al., 2008) and rat grid cells have the represen-

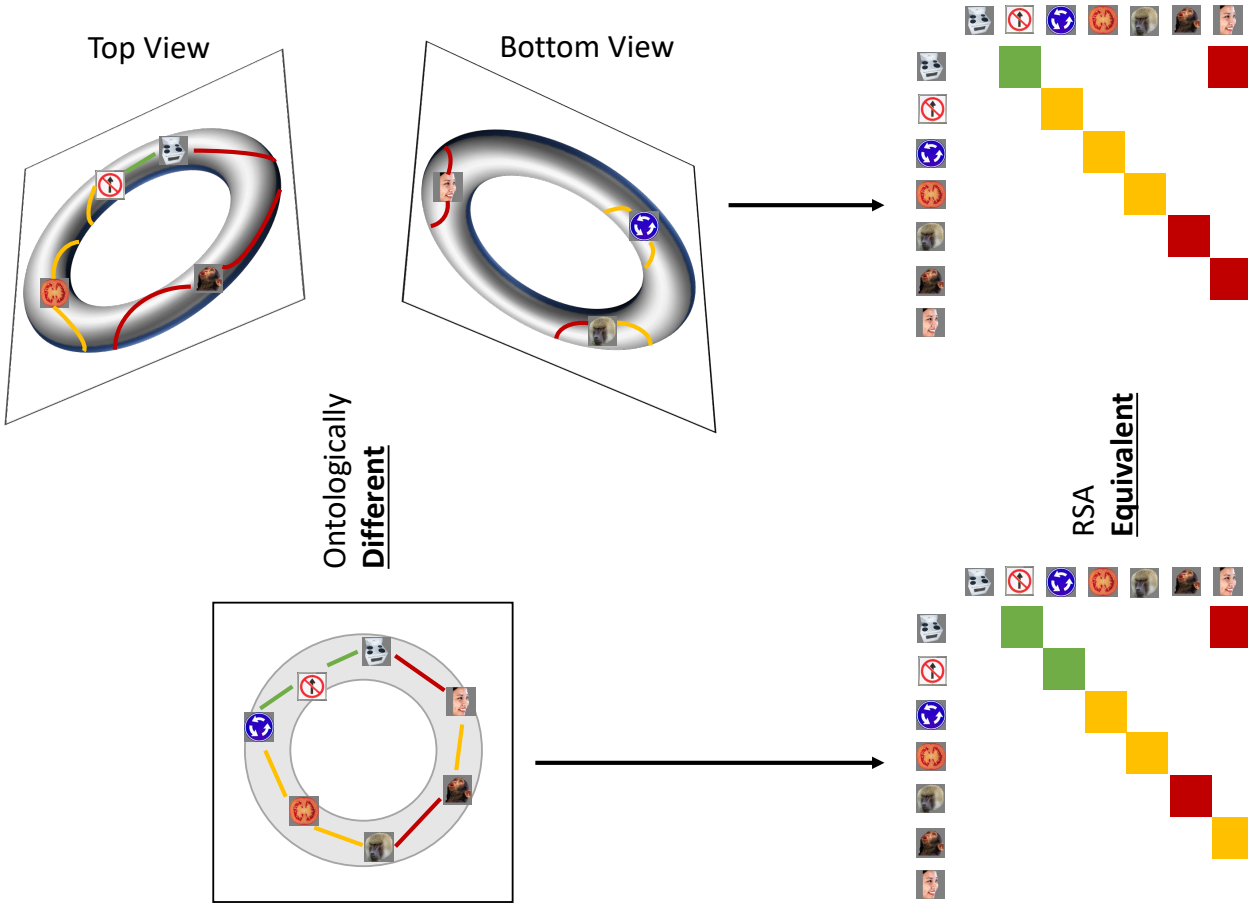

Figure 3.1: Sampled representations from a torus, projected onto an annulus, and the resulting two RDMs. The stimuli images were obtained from the supplemental information in Kriegeskorte, 2009 but were originally introduced in Kiani et al., 2007, as is the case in later figures. Top left are the top and bottom views of seven sampled representations from the surface of the torus, with colored lines indicating representational distances between adjacent points (green for small distances, yellow for medium and red for large). Bottom left is the projection of these torus representations onto an annulus, with updated representational distances. These distances for both shapes are color-coded in their respective RDMs, which would be considered equivalent by RSA, despite the representational spaces having completely different shapes.

tational space of a torus (Curto, 2017). Thus we must use tools that are sensitive to the shape of the representational space of neurophysiological data, or we may err in drawing similarities between two cells (e.g. grid cells and V1 simple cells) based on a simplified model of their representation.

But does this example demonstrate a problem using RDMs to capture representational geometry or rather a problem using correlation as a second-order isomorphism? It has previously been suggested that using non-linear second-order isomorphisms would better account for non-linear geometries (Kriegeskorte and Kievit, 2013), and some studies proposed such isomorphisms for analyzing correlation matrices of neurological data (Shahbazi et al., 2021; You and Park, 2022). A technique called distance correlation (Szekely et al., 2008) has also been shown to be a useful measure of independence in RSA model comparisons (Diedrichsen et al., 2020), being able to capture non-linear dependencies as well as linear ones. These approaches accounted for the non-independence of pairs of correlation/distance matrix entries, but in the example offered above comparing a torus and its projection onto an annulus, the difference comes from global topology, which implicates distinct causal mechanisms – the annulus contains one periodic phenomenon (captured in one loop) whereas the torus contains two – the major loop and the minor loop which bisects the tube. Such structural features are only detectable when taking into account all dissimilarities together, not just the non-independence of pairs. For example, a Gaussian-distributed cluster, and the same cluster punctured with a small hole in its center, will have similar covariance matrices despite the former being a cluster and the latter being an annulus.

Unfortunately it is not possible in RSA to segment RDMs into features (i.e., subcomponents) of their representational geometries. Multidimensional scaling (MDS) (Mead, 1992) has been used to project RDMs into low dimensions for visualization of representational spaces (Kriegeskorte, Mur, and Bandettini, 2008), but the projection dimensions are not directly interpretable and are always linear. For example, in Kriegeskorte, Mur, Ruff, et al., 2008 RSA found evidence of a shared neural code in primate IT cortex, but MDS

embeddings only revealed a blob-like distribution of representations coarsely separated by object category. If segmentation of representational spaces were possible, we could have linked representational similarity to features of the representational spaces (see below), but this is not possible with current RSA methods. In summary, RDMs, and RSA by proxy, do not address complex (i.e, global and not linear) representational geometries and the question of appropriate second-order isomorphism may only be solved once the representational geometry is appropriately captured.

The mathematical discipline concerned with studying distance/adjacency between objects in an abstract space is *topology* (Hatcher, 2002), and tools from computational topology can be applied to multivariate data to derive topological metrics. Structure or shape of data at a local scale can be integrated into global shape descriptors using topological tools, and shape features can then be quantified and analyzed to capture the richness of data structure. Topology has a number of desirable qualities for analyzing representational geometries. For example, the topology of an object does not change when the object is rotated, stretched or reflected (Hatcher, 2002). Robustness to these transformations would also be expected of representational geometries – the features of neural codes do not depend on the order of the labels of functional units, or the scale of neural activity (Laakso, 2000).

The field of *topological data analysis* (TDA) provides practical tools for analyzing data using topology, and TDA has been applied in myriad fields (Carlsson and Vejdemo-Johansson, 2021). The most established tool in TDA is *persistent homology* (PH, Edelsbrunner et al., 2000; Zomorodian and Carlsson, 2005), which seeks to identify topological features present in data, classified by their dimension. Persistent homology takes as input the distance matrix of a dataset (like an RDM), and identifies structures inherent in the data, such as clusters, loops and voids (which are elements of the H0, H1 and H2 groups respectively). Persistent homology also provides information about the the sizes and density of points belonging to these structures, which can be used to determine which features are significant.

The workflow of the PH algorithm can be seen in Figure 3.2. The process begins with a sweep through values of a linkage radius – a parameter that defines the extent of the neighbourhood within which two data points would be joined (linked) to form a structure (called the *Vietoris-Rips complex*), and the topological features of these structures at each linkage value are classified as belonging to either H0 (clusters), H1 (loops), H2 (voids), etc. As we sweep through linkage values, features will appear, persist across some range of linkage values, and then disappear – as shown in Figure 3.2, the points on the loop form that loop only within a certain range of radii B and all points eventually fully connect at radius C, destroying the loop structure. The linkage values where a feature comes into existence and ceases to exist are called the birth and death values, respectively. For further mathematical details on this process see Edelsbrunner et al., 2000; Zomorodian and Carlsson, 2005. The birth and death values, along with the feature dimensions, are plotted in persistence diagrams as shown in Figure 3.3, and points which having death value much larger than their birth value (i.e. their point in the persistence diagram is high above the diagonal line where birth and death are equal) are called "persistent". A thresholding procedure (Fasy et al., 2014) can then be used to distinguish between persistent, i.e. significant, topological features and non-persistent, i.e. noise, topological features, and an example of this can also be seen in Figure 3.3. Another useful piece of information that can be extracted for each topological feature is the *representative cycle*, which is a subset of data points in a given topological object. For additional details, see Chazal and Michel, 2017.

As the number of sampled points in an object grows, the topological features of the points, recovered by persistent homology, converge to the underlying features (Chazal et al., 2014) even in the presence of noise in the dataset (Edelsbrunner et al., 2000). In this sense, increased sampling provides greater validity of the topology of the data space.

It has been demonstrated that persistent homology can detect event-related periodic spatial signals (i.e. spatial loops) in simulated event-related fMRI data (Ellis et al., 2019). A related technique for calculating persistence diagrams called "persistent cohomology"

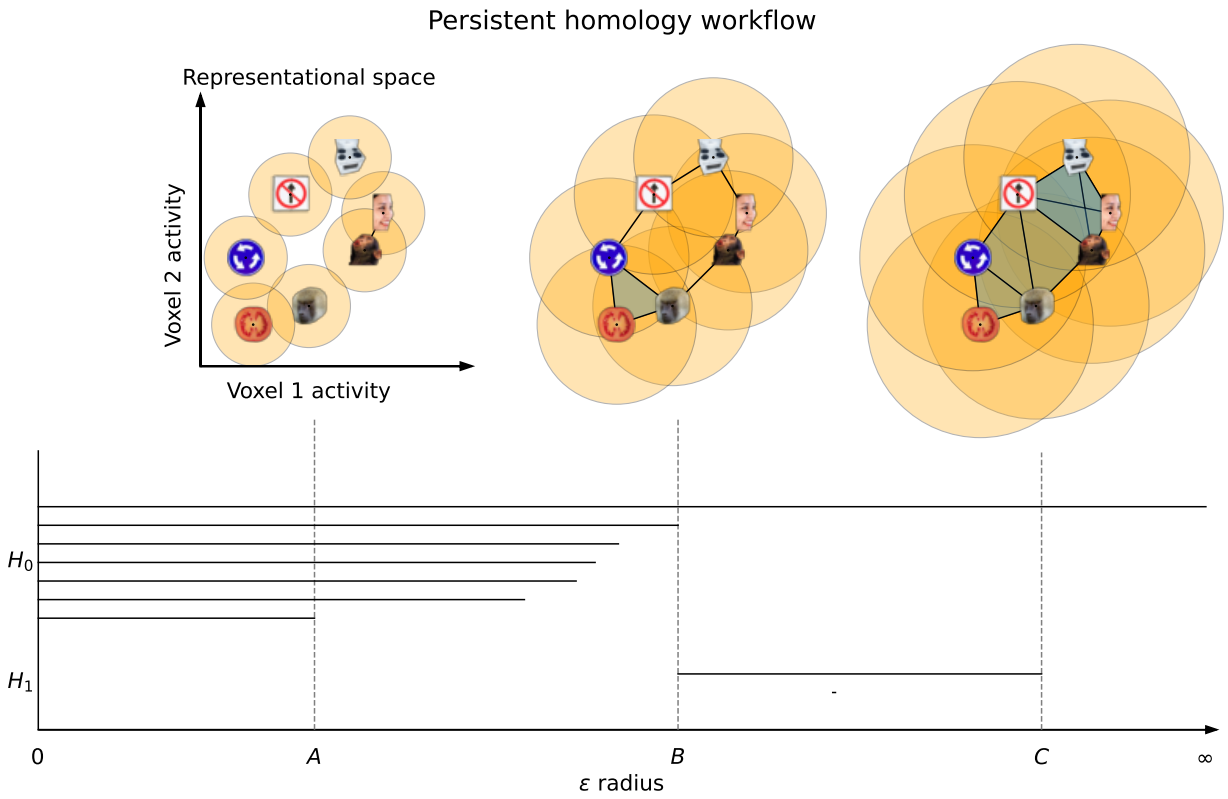

Figure 3.2: Persistent homology workflow. A linkage radius ϵ is increased from 0 and representations (i.e. data points) are connected when their distance is at most ϵ, forming Vietoris-Rips complexes. Seven clusters and two loops are present in the dataset, and are tracked by the PH algorithm with each having its own line segment. At linkage radius A there are six clusters (since the human face and monkey face are connected, and hence one cluster has died off), while at radius B the loop is fully connected (and all components merge into one) and at C the loop is filled in (i.e. is no longer a loop).

can detect representational space topologies of neural population responses of simulated rat neurons (Kang et al., 2021). Persistent homology has also been used to find meaningful structure in correlation matrices of spike trains in rat place cells using vectorized summaries of persistence diagrams called Betti curves (Giusti et al., 2015). PH also correctly characterized a low-dimensional neural manifold of mouse behavior analyzing binned spike counts of thalamic neurons (Chaudhuri et al., 2019). The results of these studies suggest that persistence diagrams are a useful tool for characterizing representational spaces, but four properties of diagrams make them particularly well-suited for this task:

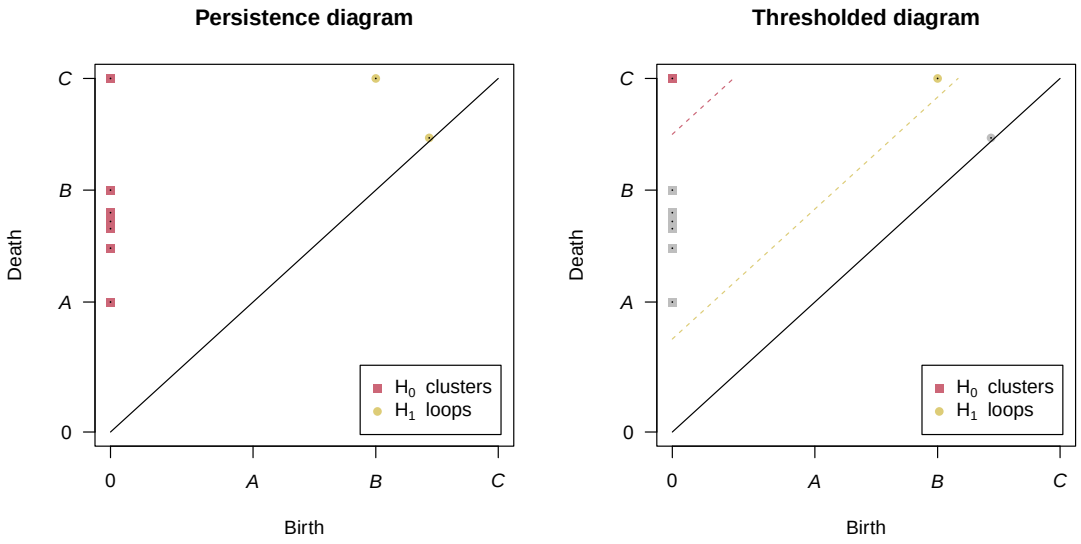

Figure 3.3: The output persistence diagram of PH run on the example dataset in Figure 3.2 (left) and an example thresholded diagram (right). In the persistence diagram there are points for each of the seven clusters and two loops - one loop is very close to the diagonal line where birth and death are the same, indicating that this loop was very "short-lived". In the thresholded diagram only one cluster and one loop were significant, indicated by their color and placement above their respective threshold lines.

1. The topological features in persistence diagrams can be identified in their input datasets, thereby allowing us to segment datasets into representational features.

2. Persistence diagrams remain consistent under different orderings of the same variables.

3. Two persistence diagrams can be meaningfully compared even if their input datasets contained different numbers of data points or variables.

4. Persistence diagrams converge as the number of data points in their input datasets grow (Chazal et al., 2014).

Property 1 means that we can uncover topological features of representational geometries, allowing for constraints on the mechanisms implicated while making comparisons

between systems more interpretable. For example, a torus and a loop have different numbers of significant loops (two and one respectively), and therefore we could distinguish between RDMs sampled from them. Also, two systems with similar linear aspects in their geometries may perform different calculations and this could be uncovered by investigating their topological sub-structures.

On the other hand, Properties 2 and 3 suggest that using persistent homology to analyze representational geometries may allow for the pooling of data from different studies, even studies with different (but relatable) sets of conditions/stimuli so long as their pooling is defensible and interpretable to the researcher – for examples, studies investigating face processing may use different face conditions, different non-face stimuli, etc, and Properties 2 and 3 of PH allow us to pool results from these studies for stronger inference of representations.

As multiple RDMs can be compared using RSA, we would need an equivalent topological tool to compare multiple persistence diagrams. For second-order isomorphisms of persistence diagrams there exist two main approaches in the literature – for differences, we can use distance calculations (Kerber et al., 2017) and for similarities we can use kernel calculations (Le and Yamada, 2018). Since topological features can be comprised of any number of data points (representations), we can capture differences between any number of data points between two representational geometries using these topological second-order isomorphisms. While in regular RSA differences and similarities are essentially opposites (like in the case of correlation and correlation distance), due to the complex nature of persistence diagrams (Turner et al., 2014) we need specialized and distinct tools to calculate their differences and similarities.

Two typical analyses of RDMs include

- Inference – deciding if two RDMs or two groups of RDMs are similar/different (an important example of which is model comparison), and
- Visualization – using MDS to project an RDM into low dimensions (Kriegeskorte, Mur, and Bandettini, 2008).

Similar analyses can be performed with persistence diagrams – differences among sets of persistence diagrams can be found using distance-based permutation approaches as in Abdallah et al., 2023; Robinson and Turner, 2017 – and the pairwise distances between multiple persistence diagrams can be used to form an MDS embedding of the diagrams into a low-dimensional space. We have implemented these analytical and inferential tools to carry out TDA on large multivariate datasets (e.g. fMRI) in our software package `TDApplied` (Brown and Farivar, 2022). Therefore, the machinery is in place to analyze persistence diagrams computed from RDMs in ways similar to RSA.

We propose a new approach called *representational topology analysis* (RTA) for detecting structures of representational space. In RTA, RDMs are converted to distance matrices (although this is not necessary for correlation dissimilarity matrices; see the methods section) and then analysed with persistent homology, resulting in *persistence diagrams*, i.e. *representational topologies*, that can then be analyzed with topological machine learning and inference methods. Comparing persistence diagrams is preferable to comparing RDMs because the latter do not encode the topology of data space, while the former explicitly represents this information. Representational topology analysis is ideal in conjunction with regular RSA (for inference on linear aspects of data space) in order to make powerful inferences about representational geometry and, by extension, fundamental mechanisms that gave rise to them. Interpretations from RTA are also complementary to interpretations from RSA because in the topological case we can make conclusions about when two representational geometries are different or similar topologically, compared to the regular RSA case where we can only say if two geometries are linearly different or similar. Below, we applied RTA on two datasets and were able to answer questions that regular RSA could not, demonstrating the potential value of representational topology.

3.5 Results

In order to compare RTA with RSA we carried out two studies – the first used data from one of the seminal studies of RSA, Kriegeskorte, Mur, Ruff, et al., 2008, and the second used data from a study of shared representations of naturalistic movie viewing across subjects, Hall, 2009; Zhang and Farivar, 2020.

3.5.1 Human and monkey IT cortex data

One of the earliest applications of RSA to visual fMRI studies was Kriegeskorte, Mur, Ruff, et al., 2008, in which RSA was used to show a common representational code in the primate inferior temporal cortex by comparing fMRI data in humans and electrophysiology data in monkeys (which was collected in the study Kiani et al., 2007). But what topological representational features exist in this shared space? Regular RSA cannot segment RDMs to find features of a representational space, and therefore cannot address this question. One of the authors of Kriegeskorte, Mur, Ruff, et al., 2008 provided us with the mean RDMs from the group of four humans and the group of two monkeys for the 92 visual stimuli displayed in Figure 3.4. The stimuli in the experiment were images of various categories, including animals, humans, body parts, naturalistic scenes and objects, and these images can be found in the supplementary data of Kriegeskorte, Mur, Ruff, et al., 2008.

A common analysis of persistence diagrams includes identifying the most persistent feature, i.e. the topological feature with greatest difference between their death and birth values – it ”lives” longest compared to all other features. We identified the most persistent loop from each of the two diagrams, which we will refer to as the ”human loop” and ”monkey loop”. For each loop we calculated a representative cycle, i.e. a subset of the 92 stimuli on each loop. Representative cycles are a useful tool for exploring topological features because those features may occupy distinct regions of the data space. For example, imagine a dataset with a loop and a cluster (not touching each other) – all the data points

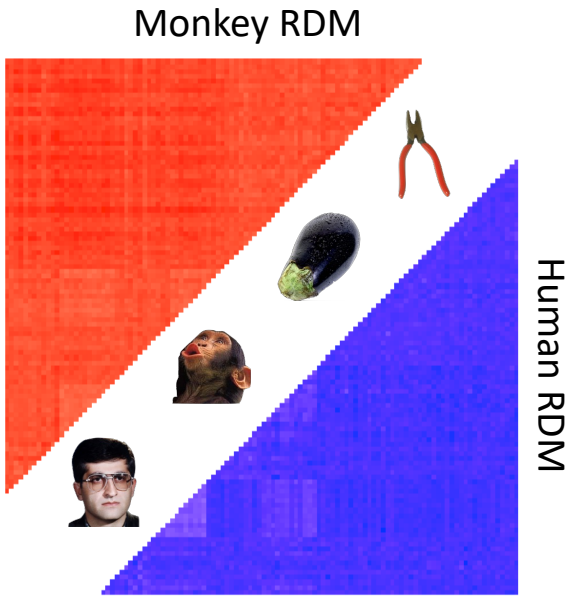

Figure 3.4: The mean human (bottom right) and monkey (top left) RDMs (each converted to a distance matrix using the transformation $1-\rho \rightarrow \sqrt{2 * (1-\rho)}$). Deeper colors indicate greater representational distances.

would be used to calculate the data's persistence diagram but only a subset of the points would lie on the loop. On the other hand, the points that do not lie on the loop, even if they are not part of any other interesting topological structure, can still have immense value in non-topological analyses (for instance in RSA analyses).

A visualization that can help identify data features defined by topological features is the graph defined by a Rips complex, called a *Vietoris-Rips graph* (Zomorodian, 2010) or VR graph for short. At each linkage radius ϵ in persistent homology, a set of edges are defined between the data points based on their distances (all the distances $\leq \epsilon$), and this defines a graph at each linkage. We visualized the VR graph of the monkey RDM at the linkage radius of the monkey loop birth, and likewise for the human RDM and loop, in Figure 3.5 to get a sense of the topological structure of the monkey and human RDMs at those linkage scales.

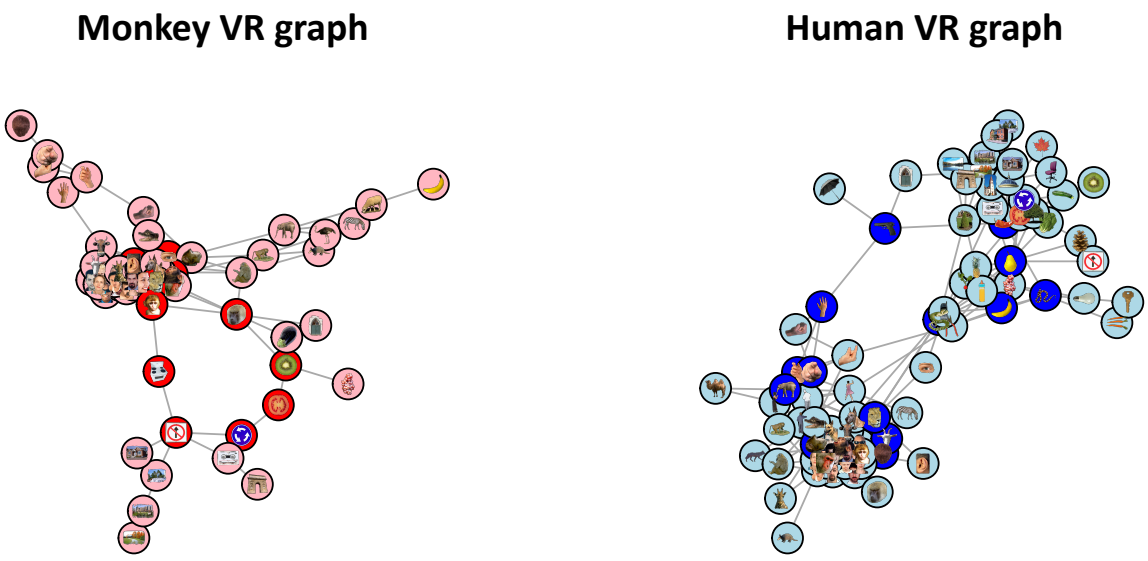

Figure 3.5: The VR graphs of the monkey RDM (left) and the human RDM (right) at the scales of their respective loop births, with the stimuli in the representative cycles of the two loops highlighted. The monkey visualization shows a central cluster of animal and monkey faces, from which the loop and two flares (an animal body flair, right, and a hand flair, top left) stem from. From the loop there is also one flair which corresponds to scenery. Only 54 of the 92 stimuli were plotted as these vertices made up the connected component of the VR graph which contained the loop (each of the other 38 stimuli either had no connections to other stimuli or formed small, topologically uninteresting clusters). The human visualization contained 81 of the 92 stimuli, and appears to be two dominant clusters with two paths of sparse connections forming the loop. The clusters are animate objects (left) and inanimate objects (right).

Striking differences occur between the two representational spaces in this view – the monkey VR graph highlights substantially more clustered representations that we can easily label, such as animals, hands, faces, objects, etc., while the human representational spaces appear to be organized into two clusters symmetrically around a loop. That in both cases the representations appear to be lobes organized around a central confluence is intriguing, and may merit greater investigation. It is worth noting that RSA suggests that the monkey and human representations are highly comparable (Kriegeskorte, Mur, Ruff, et al., 2008), finding a gross clustering into animate and inanimate objects in both

human and monkey spaces, whereas RTA reveals the ways in which they are actually different.

3.5.2 Naturalistic movie viewing data

In Zhang and Farivar, 2020, local spatial patterns of BOLD activity in subjects viewing 2D and 3D naturalistic movies (Hall, 2009) were found to be highly conserved across subjects in early visual areas and were modified by region and visual stream – early, ventral and dorsal. It would therefore be expected that group-average topological features differ by region especially for regions in different visual streams. In order to test this hypothesis we analyzed region-level data from Zhang and Farivar, 2020, constructing timepoint-by-timepoint spatial-pattern correlation distance RDMs (i.e., the correlation distance between time *i* and *j* of the BOLD patterns in each region) and used RTA to characterize the shape of representational space in group average RDMs from certain early, ventral and dorsal regions.

Movie viewing is a naturalistic task that typically induces very similar temporal patterns of activity in a group of subjects (Hasson et al., 2004a, 2010) and it has recently been shown that this similarity is likely driven by gamma oscillations (Y. Chen and Farivar, 2020) and is detectable in the spatial patterns in a manner that is more informative of viewing condition (stereoscopic 3D vs mono) than the temporal pattern correlation (Hasson et al., 2004b). Here, we used RTA to determine whether the structure of the representational space of spatial patterns over time is different between regions/streams.

To this end we computed group-average region-level significant topological features – loops which survived the thresholding procedure of Fasy et al., 2014 (see the methods section for details). Significant loops which exist at different scales (i.e. with different birth and death values) would indicate qualitatively different representational structures. We computed the mean RDMs for five regions, across subjects, of higher ventral regions VO1 and VO2, higher dorsal regions PHC1 and PHC2 and the early region V3 in both hemispheres for both 3D movie clips, resulting in 20 RDMs. We chose to analyze only

3D movie data to ensure that there was no confounding effect of stimulus condition in our analysis, and because 3D movies are closer to naturalistic stimuli than 2D movies. We used the bootstrap procedure to identify significant loops, and determined the most persistent loops from the VO regions, PHC regions and V3. The result was three group-average RDMs – one VO RDM, one PHC RDM and one V3 RDM.

In order to compare the three representational spaces, we plotted the VR graphs of the three mean RDMs, at the scale of their respective loop birth values, subsetted to contain only the data points which were in the components of their respective loop representative cycles. Since each graph represents data from one movie across subjects, each graph node represents a TR in its graph's movie, so each node is plotted with the movie frame five seconds prior to the TR (accounting for the hemodynamic lag). To determine if and where RSA provided a complementary view of these graphs, we projected the three RDMs (subsetted for the TRs in their respective VR graphs) into 2D space using MDS and colored each node in the VR graph by its location in MDS space. We call this novel visualization a *proximity-labelled rips graph* (PLRG for short). Finally, in order to link the PLRG's back to the raw data we also plotted the movie frame associated with each graph node at the node's location (in the graph space, not in the MDS space). The results can be seen in Figure 3.6.

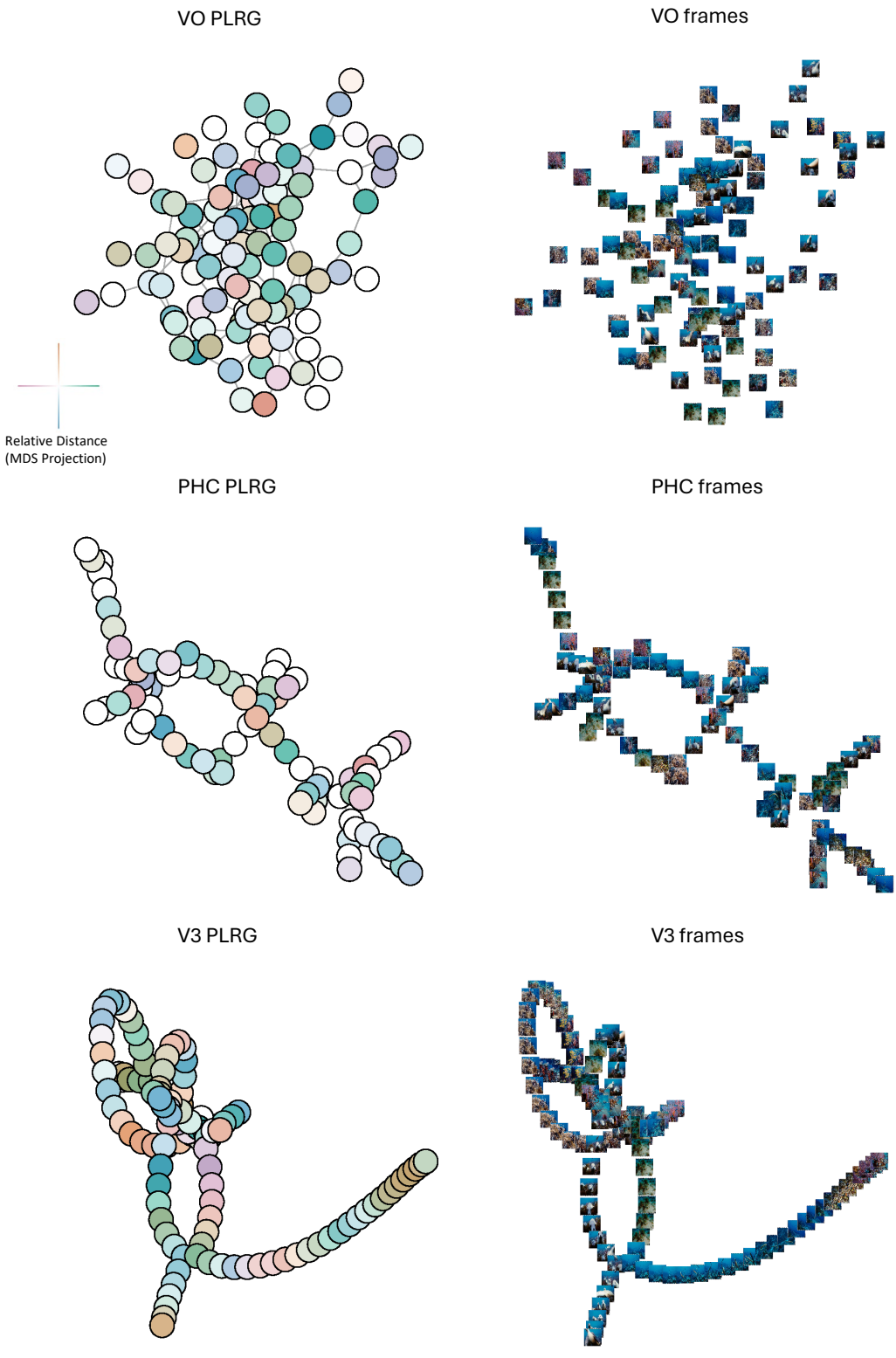

Figure 3.6: Caption on following page.

Figure 3.6: Topologies of mean representational spaces in VO (top row), PHC (middle row) and V3 (bottom row) areas. Left column is the PLRG laid out using a graph-layout algorithm, right column are the frames corresponding to each graph node, plotted at its node's 2D coordinate in the graph. The color-coding scheme for PLRG nodes, based on MDS coordinates, is displayed to the left of the VO PLRG – the x coordinate determines a horizontal color which is green for positive x-values and purple for negative x-values, and a vertical color which is orange for positive y-values and blue for negative y-values, and two nodes which have similar colors are TR's with correlated activity patterns, i.e. are nearby in MDS space. PHC and V3 have clearly-defined topologies in their PLRGs, whereas VO has mainly one densely-connected cluster. As well, the lack of color-clustering and smooth color gradients in the VO and PHC PLRG's indicate that MDS, i.e. RSA, did not capture the graph structure well. V3 on the other hand did exhibit color clustering and gradients, suggesting that there was a stronger relationship between topology and geometry at the loop birth scale. Moreover, the clustering and gradients suggest that some folding of the graph may be appropriate, where nodes which are far apart on the graph with similar colors may actually be proximal in terms of the geometry of data space. The frame visualization of V3 also appeared to most smoothly vary by color and scene type compared to PHC and VO.

The differences in the topological structure between the three regions can be readily appreciated, and these structures were not accounted for by MDS of the RDMs. This illustrates the importance of topological analysis of representational space for inference on similarity.

3.6 Discussion

We demonstrated the potential of topological analysis in identifying representational structures in stimulus-driven fMRI patterns, and showed how this knowledge of the representational geometry can be complementary to standard RSA. Importantly, sensitivity to topological features allows one to find non-linear dimensions in representational spaces, such as the loop we reported for the monkey IT data. This approach goes beyond

classic inferential statistics and allows us to have insight into the nature of the mechanisms underlying neural representations.

We first examined two RDMs, one averaged from four human's IT cortices and one averaged from two monkey's IT cortices. Unlike in Kriegeskorte, Mur, Ruff, et al., 2008 we found that the representational spaces were different because the most persistent loops in the monkey and human representational spaces did not appear to encode the same information – the monkey loop likely encoded a continuous spectrum of change in object category, whereas the human loop was more likely the distal connections between animate and inanimate clusters. Two possible explanations of the differences between the two loops could be that (1) human IT cortex efficiently resolves object category into natural and animate clusters, whereas this distinction is more blurred (i.e. continuous) in monkey IT cortex, or (2) the representational spaces are distinct simply because humans and monkeys can have very different semantic encodings of the same image. The first explanation seems unlikely – the monkey VR graph also had a clear distinction between animate and inanimate objects. The second explanation seems more likely, for example a giraffe and a monkey may have similar representations in humans because they are both animals found in Africa, and in monkeys because they are both non-dangerous creatures – in other words, there is not a one-to-one semantic correspondence. This result is perhaps the best exemplar of the major criticism of RSA described earlier – the two species may be performing very different calculations, and this difference was only detectable using RTA.

We carried out the same analytic approach to a naturalistic movie viewing dataset (Zhang and Farivar, 2020). We analyzed group-mean topological structures in VO, PHC and V3 areas. Our novel proximity-labeled rips graphs of the spaces, at the scale of their most persistent (significant) loop's birth, were visually very different between the three regions and could not be accounted for by geometry alone – the VO and PHC PLRGs did not have similar colorings of nearby nodes, and the V3 PLRG had clusters of similarly-colored nodes which existed far apart in the graph. By plotting the frames corresponding

to 5 seconds prior to each TR over each TR's graph node, we can see that V3 is likely representing low-level movie features – like object position/movement or scene color, as demonstrated by the many neighboring frames which seem to only differ in the position of objects in the frame or the scene color. On the other hand, there is not a clear division of scene (object) category in the VO PLRG, nor is there a clear relationship between graph structure and object movement/position in the PHC PLRG, but the PHC does exhibit a clear structure (as opposed to VO). In this example RTA was also able to capture aspects of representational spaces which RSA could not.

While RSA provides a "hub" for researchers to integrate data from different modalities, species, etc., it may be limited by the requirement of fixed-size matrices to encode representational geometries and correlations. Because persistent homology

- is invariant under reordering its input data,
- can compare outputs regardless of the number of points it used as input, and
- its output converges as the number of data points grows,

representational topology analysis may be well-suited to compare RDMs across RSA studies which do not have the same stimulus set or set size, building on ideas first proposed in Laakso, 2000. Concrete evidence for this use-case of RTA is hidden in our naturalistic movie analysis – free movie viewing does not follow the traditional task-based experimental design of RSA studies, which is necessary to compute stimulus-stimulus representational dissimilarities, but the flexibility of RTA allowed us to consider each time point as a "stimulus", the spatial pattern at a time point its "representation", and carry out principled comparisons of the topological structures which arose across subjects and movies. This means that fMRI datasets (as well as data from other functional neuroimaging modalities) can be compared regardless of their experimental design or duration, which would be particularly interesting for resting-state datasets. Resting state data is characterized by co-fluctuations between distal but functionally-related regions (Biswal

et al., 1995; Cordes et al., 2000; De Luca et al., 2006), which implies the existence of periodic spatio-temporal signals that could be detected with persistent homology – spherical representational topologies have already been identified in resting-state (and naturalistic image viewing) electrophysiological data from V1 in monkeys (Singh et al., 2008) and this topology could be explained by the interactions between the (periodic) orientation and spatial frequency feature maps. To our knowledge RTA is the first framework that allows comparisons between scans of different duration and study design without temporally collapsing data.

Despite the unique capabilities of RTA, it does have several limitations. Firstly, it is more complicated than regular RSA – there are more computational tools which are needed to carry out a topological analysis. Secondly, RTA is more computationally demanding – persistent homology can be computed quickly with small RDMs (up to around 100 stimuli) in low dimensions, but computing higher-dimensional homology with large RDMs will likely be slower. Similarly, the analysis procedures for persistence diagrams can take time if the persistence diagrams contain many points (although this can be remedied by using the bootstrap procedure to only select significant topological features) or if there are a large number of persistence diagrams (as in a fMRI searchlight analyses). Thirdly, RTA does need a minimum number of stimuli in an experiment to potentially be able to find meaningful topological structure – there are no formal rules, but probably at least ten to find a loop and at least twenty to find a void might be a reasonable assumption. However, in Kriegeskorte, 2009 it is suggested that regular RSA performs best when there are many stimuli, so the same would hold for RTA.

Representational topology analysis directly addresses the topology of representational space – an aspect that RSA (as a linear geometric method) cannot. This understanding of representational geometry is useful in that it can reveal non-linear dimensionality of the representation space which has direct implications for the nature of the input patterns and, by extension, the mechanisms that give rise to those input patterns. In this manner,

understanding the topology of representational space provides for novel insights unafforded by existing methods.

3.7 Materials and methods

3.7.1 Human vs. monkey comparison

We received two RDMs from the authors of Kriegeskorte, Mur, Ruff, et al., 2008, one which was the average RDM from four human subject's 3T fMRI data and the other of which was the average RDM from two monkey subject's electrode recording data. The entries of the RDMs were (average) correlation distances (i.e. 1 subtract Pearson correlation) between the spatial response patterns of voxels/cells for each pair of stimuli. For more details, see Kriegeskorte, Mur, Ruff, et al., 2008. We further transformed the correlation distance values from $1-\rho$ to $\sqrt{2(1-\rho)}$ which better satisfy the mathematical notion of distance (Brown and Farivar, 2022).

We calculated persistent homology of the two RDMs using the R package `TDA` (Fasy et al., 2021), up to homological dimension 1 (loops), up to the connectivity radius which was the maximum RDM entry, and using the `dionysus` library functionality (Morozov, 2017) to calculate representative cycles (i.e., a subset of the data points that lie on each loop) for the loops. We then computed VR graphs (Zomorodian, 2010) from the two RDMs at the scale of the birth radius of the most persistent loop for each RDM. The stimuli in the two representative cycles were highlighted with deeper colors. The layout of the graph is optimized to project connected nodes nearby each other in 2D space and unconnected nodes further apart, using a graph layout algorithm from the R package `igraph` (Csardi and Nepusz, 2006). We plotted only the graph component which contained the representative cycle nodes. The computation and visualization of the VR graphs was performed by `TDApplied`.

3.7.2 Naturalistic movie viewing study

For a detailed account of the data, acquisition and preprocessing of our naturalistic movie viewing analysis, see Zhang and Farivar, 2020. The study collected 3T fMRI data, with 3mm^3 voxels, from 55 subjects watching four 5-minute movie clips in one scan (two clips each viewed in both 2D and 3D). The TR was 2 seconds, and the first 1 minute of each movie clip was not analyzed, resulting in 120 TRs of data for each movie clip. Data preprocessing was carried out with the AFNI software (Cox, 1996) and fMRI voxel data was projected onto cortical surface nodes (36002 per hemisphere) with the SUMA (Saad and Reynolds, 2012) and FreeSurfer (Fischl et al., 2002) software packages. Cortical regional boundaries followed the probabilistic atlas from Wang et al., 2015.

We chose to solely analyze 3D movie clips in our analysis, in the regions V3, VO and PHC. To calculate an ROI RDM in a hemisphere for a particular movie clip we selected the surface nodes in that hemisphere which were in the ROI (based on atlas boundaries), and computed Pearson correlation between each pair of TRs of the time series activity of all the nodes in that movie. This resulted in a 120x120 representational similarity matrix, which was converted to an RDM by transforming each correlation value ρ to the distance value $\sqrt{2(1-\rho)}$. To obtain a group average RDM for each region, movie and hemisphere, we averaged the subject-specific RDMs.

We calculated persistent homology of the RDMs using the R package `TDAstats` (Wadhwa et al., 2019), up to homological dimension 1 (loops) and up to the connectivity radius which was the maximum RDM entry. This homology calculation was used in conjunction with the bootstrap procedure (Fasy et al., 2014) in `TDApplied` to identify significant topological features, and was implemented with 30 bootstrap iterations and significance threshold $\alpha = 0.1$ to avoid over thresholding. The subsetted persistence diagram, according to the bootstrap thresholding procedure, then contained significant group-average region-level topological features (loops). For each region – V3, VO and PHC – we identified the most persistent significant loop out of all its thresholded diagrams, the loop's birth radius and the RDM it came from. We then used the R package `TDA` to calculate

the representative cycles for those three significant loops from their respective RDMs, by performing the same persistent homology calculation with the `dionysus` library functionality.

Our novel Proximity-Labelled Rips Graph (PLRG) visualization requires an RDM, the birth scale of a loop and its representative cycle. The nodes of the PLRG graph are the TRs (i.e. spatial patterns) and connections between nodes are determined by the RDM entries which are at most the birth scale (i.e. a PLRG is a VR graph). We plotted only the graph component which contained the representative cycle nodes. Once again the position of the graph nodes in 2D were determined by the `igraph` package. In order to color the PLRG nodes, the RDM is projected into 2D using the R package `stats` (R Core Team, 2021), and the color of each node is determined by the location of its data point in MDS space according to a horizontal color scale (pink (left) to green (right)) and a vertical color scale (blue (bottom) to orange (top)). Outside of calculating the color of each node, the full visualization process of a PLRG is performed by `TDApplied`.

3.8 References

Abdallah, H., Regalski, A., Kang, M. B., Berishaj, M., Nnadi, N., Chowdury, A., Diwadkar, V. A., & Salch, A. (2023). Statistical inference for persistent homology applied to simulated fmri time series data. *Foundations of Data Science*, *5*(1), 1–25. https://doi.org/10.3934/fods.2022014

Biswal, B., Zerrin Yetkin, F., Haughton, V. M., & Hyde, J. S. (1995). Functional connectivity in the motor cortex of resting human brain using echo-planar mri. *Magnetic Resonance in Medicine*, *34*(4), 537–541. https://doi.org/https://doi.org/10.1002/mrm.1910340409

Bracci, S., & Op de Beeck, H. (2016). Dissociations and associations between shape and category representations in the two visual pathways. *The Journal of Neuroscience : the Official Journal of the Society for Neuroscience*, *36*(2), 432–444. https://doi.org/10.1523/JNEUROSCI.2314-15.2016

Brown, S., & Farivar, D. R. (2022). *Tdapplied: Machine learning and inference for topological data analysis* [R package version 0.1.0]. https://CRAN.R-project.org/package=TDApplied

Carlsson, G., & Vejdemo-Johansson, M. (2021). *Topological data analysis with applications*. Cambridge University Press. https://doi.org/10.1017/9781108975704

Chaudhuri, R., Gerçek, B., Pandey, B., Peyrache, A., & Fiete, I. (2019). The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep. *Nature Neuroscience*, *22*(9), 1512–1520.

Chazal, F., & Michel, B. (2017). An introduction to topological data analysis: Fundamental and practical aspects for data scientists. *Frontiers in Artificial Intelligence*, *4*. https://doi.org/10.3389/frai.2021.667963

Chazal, F., Glisse, M., Labruère Chazal, C., & Michel, B. (2014). Convergence rates for persistence diagram estimation in topological data analysis. *31st International Conference on Machine Learning, ICML 2014*, *1*.

Chen, X., Martin, R., & Fischer-Baum, S. (2021). Challenges for using representational similarity analysis to infer cognitive processes: A demonstration from interactive activation models of word reading. *Proceedings of the Annual Conference of the Cognitive Science Society*, *43*. https://par.nsf.gov/biblio/10321541

Chen, Y., & Farivar, R. (2020). Natural scene representations in the gamma band are prototypical across subjects. *NeuroImage*, *221*, 117010. https://doi.org/https://doi.org/10.1016/j.neuroimage.2020.117010

Connolly, A. C., Guntupalli, J. S., Gors, J., Hanke, M., Halchenko, Y. O., Wu, Y.-C., Abdi, H., & Haxby, J. V. (2012). The representation of biological classes in the human brain. *Journal of Neuroscience*, *32*(8), 2608–2618. https://doi.org/10.1523/JNEUROSCI.5547-11.2012

Cordes, D., Haughton, V. M., Arfanakis, K., Wendt, G. J., Turski, P. A., Moritz, C. H., Quigley, M. A., & Meyerand, M. E. (2000). Mapping functionally related regions of brain with functional connectivity mr imaging. *American Journal of Neuroradiology*, *21*(9), 1636–1644.

Cox, R. W. (1996). Afni: Software for analysis and visualization of functional magnetic resonance neuroimages. *Computers and Biomedical Research*, *29*(3), 162–173. https://doi.org/https://doi.org/10.1006/cbmr.1996.0014

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, *Complex Systems*, 1695. https://igraph.org

Curto, C. (2017). What can topology tell us about the neural code. *Bulletin of the American Mathematical Society*, *54*(1), 63–78.

De Luca, M., Beckmann, C., De Stefano, N., Matthews, P., & Smith, S. (2006). Fmri resting state networks define distinct modes of long-distance interactions in the human brain. *NeuroImage*, *29*(4), 1359–1367. https://doi.org/https://doi.org/10.1016/j.neuroimage.2005.08.035

Diedrichsen, J., Berlot, E., Mur, M., Schütt, H., & Kriegeskorte, N. (2020). Comparing representational geometries using the unbiased distance correlation. *arXiv*.

Dujmović, M., Bowers, J. S., Adolfi, F., & Malhotra, G. (2022). The pitfalls of measuring representational similarity using representational similarity analysis. *bioRxiv*, *1*. https://doi.org/10.1101/2022.04.05.487135

Edelsbrunner, H., Letscher, D., & Zomorodian, A. (2000). Topological persistence and simplification. *Discrete & Computational Geometry*, *28*, 511–533.

Ellis, C., Lesnick, M., Henselman, G., Keller, B., & Cohen, J. (2019). Feasibility of topological data analysis for event-related fmri. *Network Neuroscience*, *3*, 1–12. https://doi.org/10.1162/netn_a_00095

Fasy, B., Kim, J., Lecci, F., Maria, C., Millman, D., & Rouvreau, V. (2021). *Tda: Statistical tools for topological data analysis* [R package version 1.7.7]. https://CRAN.R-project.org/package=TDA

Fasy, B., Lecci, F., Rinaldo, A., Wasserman, L., Balakrishnan, S., & Singh, A. (2014). Confidence sets for persistence diagrams. *The Annals of Statistics*, *42*, 2301–2339.

Fischl, B., Salat, D., Busa, E., Albert, M., Dieterich, M., Haselgrove, C., Kouwe, A., Killiany, R., Kennedy, D., Klaveness, S., Montillo, A., Makris, N., Rosen, B., & Dale, A. (2002). Whole brain segmentation: Automated labeling of neuroanatomical structures in the human brain. *Neuron*, *33*, 341–55. https://doi.org/10.1016/S0896-6273(02)00569

Giusti, C., Pastalkova, E., Curto, C., & Itskov, V. (2015). Clique topology reveals intrinsic geometric structure in neural correlations. *Proceedings of the National Academy of Sciences*, *112*(44), 13455–13460. https://doi.org/10.1073/pnas.1506407112

Hall, H. (2009). Under the sea 3d.

Hasson, U., Malach, R., & Heeger, D. J. (2010). Reliability of cortical activity during natural stimulation. *Trends in Cognitive Sciences*, *14*(1), 40–48.

Hasson, U., Nir, Y., Levy, I., Fuhrmann, G., & Malach, R. (2004a). Intersubject synchronization of cortical activity during natural vision. *Science*, *303*(5664), 1634–1640. https://doi.org/10.1126/science.1089506

Hasson, U., Nir, Y., Levy, I., Fuhrmann, G., & Malach, R. (2004b). Intersubject synchronization of cortical activity during natural vision. *Science*, *303*(5664), 1634–1640. https://doi.org/10.1126/science.1089506

Hatcher, A. (2002). *Algebraic topology*. Cambridge University Press.

Kang, L., Xu, B., & Morozov, D. (2021). Evaluating state space discovery by persistent cohomology in the spatial representation system. *Frontiers in Computational Neuroscience*, *15*. https://doi.org/10.3389/fncom.2021.616748

Kerber, M., Morozov, D., & Nigmetov, A. (2017). Geometry helps to compare persistence diagrams. *ACM Journal of Experimental Algorithmics*, *22*. https://doi.org/10.1145/3064175

Kiani, R., Esteky, H., Mirpour, K., & Tanaka, K. (2007). Object category structure in response patterns of neuronal population in monkey inferior temporal cortex [PMID: 17428910]. *Journal of Neurophysiology*, *97*(6), 4296–4309. https://doi.org/10.1152/jn.00024.2007

Kriegeskorte, N., Mur, M., & Bandettini, P. (2008). Representational similarity analysis – connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, *2*, 4. https://doi.org/10.3389/neuro.06.004.2008

Kriegeskorte, N., Mur, M., Ruff, D. A., Kiani, R., Bodurka, J., Esteky, H., Tanaka, K., & Bandettini, P. A. (2008). Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron*, *60*(6), 1126–1141.

Kriegeskorte, N. (2009). Relating population-code representations between man, monkey, and computational models. *Frontiers in Neuroscience*, *3*. https://doi.org/10.3389/neuro.01.035.2009

Kriegeskorte, N., & Diedrichsen, J. (2019). Peeling the onion of brain representations [PMID: 31283895]. *Annual Review of Neuroscience*, *42*(1), 407–432. https://doi.org/10.1146/annurev-neuro-080317-061906

Kriegeskorte, N., & Kievit, R. A. (2013). Representational geometry: Integrating cognition, computation, and the brain. *Trends in Cognitive Sciences*, *17*(8), 401–412. https://doi.org/https://doi.org/10.1016/j.tics.2013.06.007

Laakso, A. (2000). Content and cluster analysis: Assessing representational similarity in neural systems. *Philosophical Psychology*, *13*.

Le, T., & Yamada, M. (2018). Persistence fisher kernel: A riemannian manifold kernel for persistence diagrams. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/959ab9a0695c467e7caf75431a872e5c-Paper.pdf

Mead, A. (1992). Review of the development of multidimensional scaling methods. *Journal of the Royal Statistical Society. Series D (The Statistician)*, *41*(1), 27–39.

Morozov, D. (2017). *Dionysus is a c++ library for computing persistent homology*. https://mrzv.org/software/dionysus2/

R Core Team. (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/

Robinson, A., & Turner, K. (2017). Hypothesis testing for topological data analysis. *Journal of Applied and Computational Topology*, *1*.

Saad, Z. S., & Reynolds, R. C. (2012). Suma [20 YEARS OF fMRI]. *NeuroImage*, *62*(2), 768–773. https://doi.org/https://doi.org/10.1016/j.neuroimage.2011.09.016

Shahbazi, M., Shirali, A., Aghajan, H., & Nili, H. (2021). Using distance on the riemannian manifold to compare representations in brain and in models. *NeuroImage*, *239*, 118271. https://doi.org/https://doi.org/10.1016/j.neuroimage.2021.118271

Shepard, R. N., & Chipman, S. (1970). Second-order isomorphism of internal representations: Shapes of states. *Cognitive Psychology*, *1*(1), 1–17. https://doi.org/https://doi.org/10.1016/0010-0285(70)90002-2

Singh, G., Mémoli, F., Ishkhanov, T., Sapiro, G., Carlsson, G., & Ringach, D. (2008). Topological analysis of population activity in visual cortex. *Journal of Vision*, *8*, 11.1–18. https://doi.org/10.1167/8.8.11

Szekely, G., Rizzo, M., & Bakirov, N. (2008). Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, *35*. https://doi.org/10.1214/009053607000000505

Turner, K., Mileyko, Y., Mukherjee, S., & Harer, J. (2014). Frechet means for distributions of persistence diagrams. *Discrete & Computational Geometry*, *52*(1), 44–70.

Wadhwa, R., Dhawan, A., Williamson, D., & Scott, J. (2019). *Tdastats: Pipeline for topological data analysis* [R package version 0.4.1]. https://github.com/rrrlw/TDAstats

Wang, L., Mruczek, R. E. B., Arcaro, M., & Kastner, S. (2015). Probabilistic maps of visual topography in human cortex. *Cerebral Cortex*, *25 10*, 3911–31. https://api.semanticscholar.org/CorpusID:206372126

You, K., & Park, H.-J. (2022). Geometric learning of functional brain network on the correlation manifold. *Scientific Reports*, *12*(1), 17752. https://doi.org/10.1038/s41598-022-21376-0

Zhang, A., & Farivar, R. (2020). Intersubject spatial pattern correlations during movie viewing are stimulus-driven and nonuniform across the cortex. *Cerebral Cortex Communications*, *1*(1). https://doi.org/10.1093/texcom/tgaa076

Zomorodian, A. (2010). The tidy set: A minimal simplicial set for computing homology of clique complexes. *Proceedings of the Twenty-Sixth Annual Symposium on Computational Geometry*, 257–266. https://doi.org/10.1145/1810959.1811004

Zomorodian, A., & Carlsson, G. (2005). Computing persistent homology. *Discrete and Computational Geometry*, *33*, 249–274. https://doi.org/10.1007/s00454-004-1146-y

Chapter 4

Discussion and future directions

In this dissertation I have demonstrated a new limitation of RSA – an ability to detect or compare topological structures in representational geometries – and I have developed a novel framework, called RTA, which can be used in conjunction with my new software package, `TDApplied`, to overcome this limitation. It was previously known that two systems performing distinct calculations could be erroneously equated by RSA, but I have shown that RTA is able to detect and visualize these computational differences, in a number of cases, by probing the topology of representational geometry. RTA captures representational topology by computing persistence diagrams from RDMs, and can compare persistence diagrams with distance and kernel functions in order to carry out typical RSA-like analyses. I utilized RTA to analyze the data from two neuroimaging studies – an RSA study of primate IT cortex responses to objects of various categories and a spatial-pattern analysis study of human cortical responses to naturalistic movie viewing in 3D.

In my two RTA analyses I have shown that RTA, unlike RSA, can segment representational geometries into significant (non-linear) features of neural computation. My first analysis found a novel distinction (an object-category loop in monkeys) between the functional architectures of human and monkey IT cortex while capturing, as well, expected object-category clustering that was found in previous studies. My second analysis demonstrated the utility of my new visualization technique for representational topolo-

gies called the Proximity Labelled Rips Graph (PLRG). The PLRG is a powerful tool because it directly shows *where* in a dataset representational geometry can and cannot account for representational topology. The original study found evidence of similar (i.e. shared) spatial patterns across subjects during movie viewing, to different extents in different brain regions. While these findings suggest consistent region-level computations across subjects, which vary by region, I have shown, using PLRGs, that there are region-level topological signatures of neural response that vary by region during naturalistic movie viewing.

Both of the two RTA studies relied in an essential way on my new software package `TDApplied` to carry out calculations and visualizations, and I have demonstrated the utility of `TDApplied` for topological analyses of data in a number of software documentation texts called vignettes. By analyzing data from the Human Connectome Project in a RTA-like pipeline, I have demonstrated that `TDApplied` can capture topological features of neural computation which are related to task and behavior. With a number of simulations I have also showed that `TDApplied` is more highly optimized, flexible and computationally correct compared to other R packages for analyzing persistence diagrams.

Taken together, my two RTA studies and my `TDApplied` vignettes show that the topology of representational geometry, captured by RTA and not by RSA, is critical to making interpretable and trustworthy representational comparisons. RTA should therefore direct future RSA studies to reflect more deeply on what neural features of computation are hidden within their RDMs and to explain their RDM comparisons. Future studies should validate data pooling applications of RTA, such as combining data from studies with related but different stimuli or combining data from resting-state scans, in order to produce comparisons which were previously impossible and therefore provide new inferences of neural function. RTA would be a valuable addition to any neuroscientist's toolbox.

4.1 Implications and origins of representational topologies

In Chapter 1 I discussed two concrete examples from the literature of representational topologies which exist in neurological data. First, the representational space of neural activity in primary visual cortex lives on a sphere (Singh et al., 2008), perhaps due to the interplay between orientation and spatial frequency maps. Second, in Curto, 2017, a torus representational topology was found to be a good model of rat grid cell responses because of the way their receptive fields tile the rat's field of view. From a mathematical perspective, complex topological structures can arise from gluing points on a shape – a loop is a line with its endpoints glued, a torus is a gluing of the edges of a square (Hatcher, 2002) or a hexagon (Curto, 2017). This perspective provides the simple explanation that representational topologies can arise due to same representations of distinct stimuli. For example, if we were conducting a study of orientation-selectivity in V1 and our stimuli were bars of all angle rotations, then our stimulus space would be spanned by a single angle variable which could take any numeric value. Clearly, any two stimuli whose angle differs by a non-zero multiple of 180 degrees are visually the same, and will therefore evoke the same neural representations. Therefore the representational space of all oriented bars will be glued together at angles which differ by multiples of 180 degrees, forming a loop. In this way biology takes into account the topology of stimulus features.

As noted in Chapter 1, two of the main application areas of persistent homology in fMRI analyses are analyzing connectome graphs and neural codes, and the interpretation of topological features which arise in both domains must be treated separately. One perspective on topological features of connectome graphs is that holes may represent parallel processing strategies – the divergence and later re-convergence of flows of information in the brain (Sizemore et al., 2018), a plausible theory given that the primate brain is known to employ a number of parallel processing strategies (Nassi and Callaway, 2009). Unfortunately for this theory, persistent homology forms undirected relationships between functional units so there is no notion of directed temporal flow of information when calcu-

lating persistent homology of a connectome graph. A more conservative interpretation of loops in a functional connectivity connectome graph would be a network of regions that co-fluctuate according to a single latent variable. Similarly, in the case of spatial loops in fMRI data (i.e. neural codes) defined by spatial pattern correlations, topological features do not code for temporal information. A more conservative interpretation of a spatial pattern topological features would be a collection of functional units (perhaps forming a functional network) which were not involved in neural function in certain data segments (for example the suppression of a resting-state network during the computation of a particular task). Modelling and visualization techniques like those employed in the HCP analysis in Chapter 2 can be used to help determine when topological features, like loops, are traversed temporally according to stimulus conditions or according to physiological measures like respiration. However, in the case of connectome graphs or neural codes, specific methodology should be developed for the integration of directed relationships between functional units to capture temporal patterns in topological features.

One area of neuroimaging that could provide useful ideas in this task is called "walks on neural manifolds." A neural manifold is defined as the structure of permissible neural states, and walks are the temporal traversal of these structures during neural computation (Pillai and Jirsa, 2017; Shine et al., 2019). The topological signature of a representational geometry, i.e. the collection of all significant topological features and their arrangement together in representational space, can therefore be interpreted as a neural manifold with added information about latent variables in each data segment. The study Chaudhuri et al., 2019 defined this neural manifold using persistent homology to characterize the dimension of the representational geometry (using Betti numbers) and fit splines of corresponding dimension to the data. However, this approach is different to RTA because it is not designed for comparisons across datasets, but rather for finding a latent variable representation of a single dataset. Ideas from the theory of walks on neural manifolds could provide a future avenue for incorporating temporal information from neuroimaging studies into RTA analyses.

Implicit to all of the RTA analyses in this thesis, and to the studies of neural manifolds and structured flows on manifolds, is the famous manifold hypothesis – that the essential structure of high-dimensional data is low-dimensional. We only analyzed clusters and loops in our RTA analyses, i.e. homological dimensions 0 and 1, but it could certainly be the case that higher-dimensional topological features exist in representational spaces. For example, the spherical representational space of V1 (Singh et al., 2008) and the torus representational space of rat grid cells (Curto, 2017) each contain a void – a feature of homological dimension 2. From a computational standpoint, calculating persistence diagrams in high dimensions can quickly become intractable depending on the number of representations (Somasundaram et al., 2021; Zomorodian and Carlsson, 2005). In Chapter 2 we showed that `TDApplied` is a highly efficient persistent homology engine compared to other R packages, and we would expect even more significant performance gains in higher dimensions; however, high-dimensional topological structures, in representational spaces with many points, should only be sought when there is strong evidence suggesting that such structures exist in the data.

It is important to note that a single RTA analysis is only observational. That is to say, that if we have the hypothesis that a particular representational topological feature represents certain stimulus features (for example the object-category loop identified in Chapter 3), then we could test that hypothesis in a follow-up study. The procedure would be to mix the stimuli on the topological feature, based on the hypothesized stimulus features, and seeing if the representations of these new stimuli fill in the original topological feature. This would validate not just the existence of a topological feature (which we can already do using the bootstrap procedure (Fasy et al., 2014)) but that it captures the stimulus features we think it does.

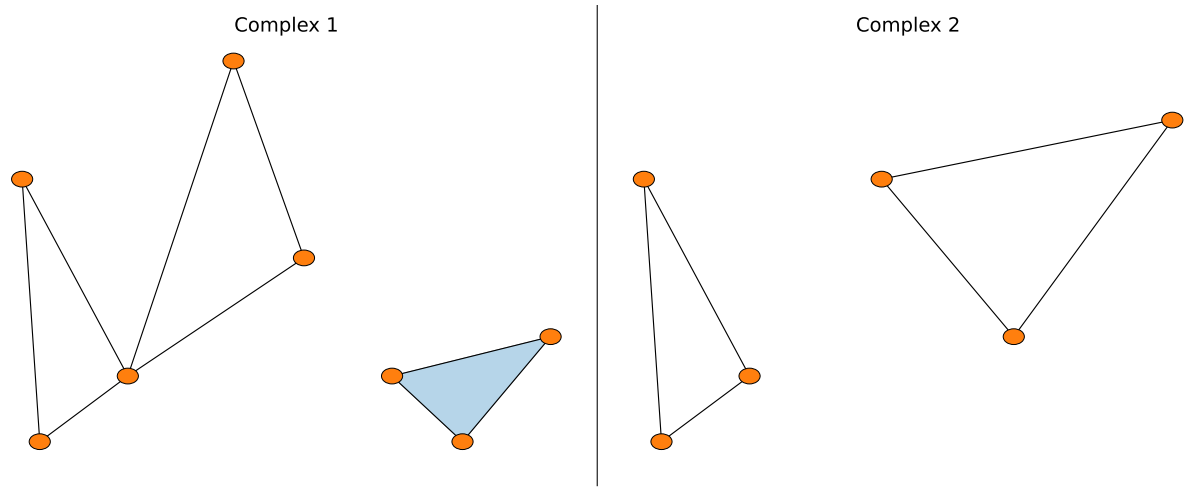

Figure 4.1: Two distinct simplicial complexes with the same homology. Both datasets have two connected components and two loops, but in the first dataset the loops are connected and in the second dataset they are not connected.

4.2 Linking topological features across multiple representational topologies

As mentioned in Chapter 1, if two shapes have different homology then they are not the same shape, but the inverse statement is not true – it is possible for different shapes to have the same homology (i.e. counts of topological features), and in the case of RTA this means that topological similarity of two neural systems does not guarantee similarity of computations (more on this in the next section). For example, the two simplicial complexes in Figure 4.1 have the same number of H_0 components and H_1 loops, but their arrangement in space implies possibly distinct generating mechanisms. The connected loops in the first complex could represent solely vertical or horizontal rotations of a 3D object – 0 horizontal rotation and 0 vertical rotation are the same transformation, but all other transformations are distinct, being either purely horizontal or vertical. The two distinct loops in second complex could represent rotation of two distinct groups of images, say of human faces and animal faces.

Visualizations like VR graphs, coupled with the information in representative cycles, can be helpful in describing the arrangement of topological features in their representational space, as in our analyses in Chapters 2 and 3, but these approaches would not be as useful in higher dimensions. It was even noted in Sizemore et al., 2019 that there is not currently a best method for picking representative cycles, which presents a challenge for comprehensively segmenting representational geometries.

Ideally there would be a method which could (semantically) link topological features across representational spaces, replacing analyses of individual features, but unfortunately no such method exists in the applied topology literature. The essential challenge is that a topological feature can not be linked to a unique subset of its representational space (for example, many collections of stimuli may capture the loop within an annulus structure). Other neuroimaging analysis frameworks, for example RSA and functional connectivity, circumvent this issue by establishing a direct correspondence between the dimensions of two data spaces (RDMs and functional units, respectively). One paper (Salch et al., 2021) proposed a method for linking loops according to their location in a shared data space (which was not the case for our analyses in Chapter 3) by quantifying translations between loops, but there are other transformations of topological features that could capture their semantic differences.

One technique that may be particularly well-suited for capturing semantic differences between topological features of the same dimension, regardless of the representational spaces they live in, is the Procrustes analysis Andreella et al., 2023. A Procrustes analysis can optimally project multiple matrices (like RDMs) into a shared data space using linear operations (like translations, scaling, rotations and reflections). The main challenge which would need to be resolved for Procustes analyses of topological features is that topological features form subsets of representational spaces, possibly resulting in different-sized matrices, which Procrustes is not designed to accommodate. Nevertheless, such an approach for linking topological features across datasets would be a useful future direction of both RTA and applied topology research.

4.3 Linear RSA compared to non-linear RTA representational comparisons

Non-linear isomorphisms for RSA have been shown to be effective tools for geometric comparison (Diedrichsen et al., 2020; Shahbazi et al., 2021; You and Park, 2022), and RTA implements only non-linear comparisons of representational topologies. On the other hand, the simplicity and interpretability of linear methods like Spearman correlation remains a convincing argument for their use in RSA experiments. I contend that neither approach – linear comparisons in RSA or non-linear comparisons in RTA – is universally better than the other, but both are necessary to make principled comparisons. That is why I suggest using RTA with RSA as complementary techniques. To argue this point, we will now provide examples of situations where we would expect linear RSA comparisons to be more effective than non-linear RTA comparisons, and vice versa.

First, let us suppose that two representational geometries were each fully captured by a single Gaussian cluster. Then the principle components of the clusters would represent linear dimensions which fully capture all of the data variance. Linear comparisons in RSA would essentially compare how well the linear dimensions of the two clusters align (possibly taking into account the variance of each dimension, depending on which second-order isomorphism is used). On the other hand, the non-linear comparisons in RTA could only determine that each dataset was comprised of a single cluster. In this case, linear comparisons in RSA would provide more detailed comparisons. However, this is predicated on the knowledge that each representational geometry was a single Gaussian cluster – we could determine the clustering structure either with MDS in RSA or with RTA.

If the representational spaces had more than one cluster, or if the clusters had non-Gaussian (i.e. non-linear) shape then the example starts to fall through for linear comparisons. In the former case, representational dissimilarities between points in distinct Gaussian clusters with distinct principle components (i.e. directions of variance) are not

informative – the number of clusters are meaningful, and components within each cluster are meaningful, but comparing points between clusters is not (since the clusters may be representing different representational features). In this case, only within-cluster representational dissimilarities are helpful for comparison, making methods like Spearman correlation of RDMs sub-optimal. RTA would be able to correctly identify the number of clusters, but again would disregard the geometric information contained in the within-cluster representational dissimilarities, so segmenting the representational geometries into the clusters and analyzing clusters independently with linear methods may be the best approach for comparison. If the clusters are non-Gaussian then linear dimensions would not be appropriate to capture data variance. For instance a cluster which is a loop is mechanistically different from a Gaussian cluster, even if the principle (linear) dimensions of variance are the same. That is not to say that RTA comparisons are better than linear RSA comparisons for *all* cases of non-Gaussian cluster comparisons, but certainly RTA will be best when clusters have some non-trivial topological structure (like loops, voids, etc.).

In Chapter 3 I provided several examples of representational comparisons which were enhanced by comparisons of representational topologies compared to linear geometric comparisons, exactly because the spaces contained some significant topological features. Based on the discussion above, single Gaussian clusters may be most appropriate for analysis with linear RSA comparisons, but what are we to do when we don't know the structure of our representational geometries? One possible future extension of this thesis work would be to combine RSA and RTA comparisons by first (1) segmenting the representational geometry into representational features (including components) using VR graphs and representative cycles and then (2) compare each distinct topological feature in one representational geometry with each distinct component in the other geometry (with both features having the same topological dimension) using linear RSA comparisons. Such an approach, as opposed to simply applying both RSA and RTA to a dataset,

would integrate topological and geometric comparisons to incorporate both big-picture and fine-scale information in comparisons, and should be pursued as future work.

Perhaps a simpler way in which linear RSA comparisons and non-linear RSA comparisons could be combined in a single framework would be forming a joint similarity function. In RSA, Spearman correlation quantifies similarity between two RDMs with a value between -1 and 1, and in RTA the persistence Fisher kernel quantifies similarity between two persistence diagrams with a value between 0 and 1. To form a joint similarity function between two representational geometries we could form a weighted average between the Spearman correlation (linearly scaled to be between 0 and 1) of the two RDMs and the persistence Fisher kernel of the two persistence diagrams. While such a function would not itself technically be a kernel function, it would still be a similarity function which takes into account geometric and topological similarity into a single score for comparison.

4.4 Model adjudication

One of the most important features of RSA is the ability to carry out model adjudication (i.e. comparison). The correlation value between two RDMs can be bootstrapped to compute a confidence interval Kriegeskorte et al., 2008, and multiple correlation values can be compared to determine which RDM out of a list of possible candidates is the best ”model” of a target RDM. We can even determine the statistical significance of a single (Spearman) correlation value, because a fisher-transformed correlation has a known, closed-form null distribution Fisher et al., 1936.

While model adjudication was not a focus of the RTA studies presented in Chapter 3, it is clear that RTA does have the potential to carry out some of these desirable procedures. For instance, given two RDMs and their respective persistence diagrams, we could generate a confidence interval for their distance/kernel value by (1) bootstrap resampling the two RDMs, (2) computing their two ”bootstrapped” persistence diagrams,

and (3) computing the distance/kernel value between the two bootstrapped diagrams. By repeating this process many times we could estimate the sampling distribution of the distance/kernel value directly, and therefore determine a confidence interval for the estimate. On the other hand, multiple distance/kernel values can be compared to determine the best candidate persistence diagram, out of a list of diagrams, for a target persistence diagram. Both of these routines should be included in future RTA studies.

Unfortunately it is not presently possible to compute the p-value for a single distance/kernel value between persistence diagrams. Therefore, even though we could identify the best candidate persistence diagram to a target diagram, it would not be clear if there was a significant relationship between the two diagrams. One possible framework for testing the relatedness of two persistence diagrams (as opposed to determining the significance of a distance/kernel value) would be to (1) compute the difference RDM of their two RDMs, (2) translating off-diagonal distances to be positive, and (3) performing the bootstrap procedure (Fasy et al., 2014). If the bootstrap found any significant topological features in the difference RDM then we would say that the RDMs were statistically distinct – otherwise similar. For RTA to have useful model adjudication machinery, a procedure (like the one described above, or one that could estimate the p-value of a single distance/kernel value) should be introduced and validated.

4.5 Data pooling of RSA studies and non-RSA studies

One of the most significant challenges in neuroimaging studies, like most RSA studies, is obtaining sufficient sample size to make inferences due to the high costs of obtaining high-quality neuroimaging data. Each RSA dataset will have its own set of stimuli, conditions, experimental design, acquisition etc., resulting in some number of stimulus-stimulus RDMs. Since the orientation selectivity of V1 neurons should be detectable regardless of which exact stimulus is being rotated (for example a bar, a squiggly line, a snake, etc.) any RSA study of orientation selectivity in V1, regardless of stimuli used,

acquisition, study design etc., should result in representational geometries defined by a loop. These kinds of cross-dataset comparisons are not possible in RSA, but are possible in RTA. In fact, we can compare data from any two RSA studies, although when there is a relationship between the two sets of stimuli (based on the questions the studies were trying to answer), for example, two sets of images of faces and houses, this type of comparison would be most useful. Interest in these types of comparisons were discussed in Laakso, 2000, but have lacked a principled framework for their application.

RSA studies use a task-based neuroimaging paradigm to compare stimulus representations, but this is not the only kind of neuroimaging experimental framework. Hidden in my HCP and natural movie viewing analyses in Chapters 2 and 3 is a new avenue for application of RTA to resting-state neuroimaging analyses, or to studies with different (but ideally related) study designs. In both analyses I computed RDMs as correlation distances of spatial patterns over time, considering each time point as its own stimulus and each spatial pattern as its representation. Since RTA can compare persistence diagrams regardless of the number of data points in the underlying representational spaces, RTA could therefore compare persistence diagrams computed from resting-state data of any duration, or between task-based functional data regardless of duration or study design. The case of resting-state topological comparisons would be very interesting – the non-independence of (periodic) activity of resting-state networks may be captured in low-dimensional topological features which would define new neural mechanisms at rest. On the other hand, comparisons of studies with different designs would also be extremely interesting – the benefits of naturalistic neural stimulation from naturalistic movie viewing could be combined with the benefits of targeted task-based neural signals evoked by carefully engineered image stimuli to make new and exciting inferences about neural function using RTA.

4.6 Final conclusions and contributions to knowledge

The topology of representational geometry captures meaningful features of neural computation that are missed by RSA. In this thesis I have demonstrated that:

1. RTA can analyze persistence diagrams, by using `TDApplied`, in ways that RSA can analyze RDMs.

2. RTA can segment representational geometries (i.e. RDMs) into topological features of neural computation, which can be visualized to make principled and interpretable comparisons of representational geometries.

3. PLRGs can locate where in a representational dataset geometry does not account for topological features of neural function.

4. `TDApplied` can be used to identify and interpret task and behavior-related features of neural computation.

5. `TDApplied` is a state-of-the-art R package for efficient and flexible topological analyses of data (a suite of analyses which cannot currently be carried out by other packages).

The ability to capture, compare and analyze (with machine learning and inference) representational topologies has significant implications for future research. Future RSA studies can compare representational geometries as well as representational topologies, with RTA, to make more trustworthy inferences. By segmenting RDMs into topological features, future studies can also link abstract representational dissimilarities back to structures/patterns of representations, tying together topological features, neural features of computation and stimulus features. Due to the flexibility of persistent homology, data from RSA studies which were previously not comparable can now be compared, meaning higher statistical power in making inferences and being able to answer novel questions of

neural function. Finally, since persistence diagrams are used in applications in many domains (i.e. outside of neuroimaging) our software opens the door to interpretable, flexible and powerful applied topological analyses to the wider research community.

4.6 References for Chapters 1 and 4

Abdallah, H., Regalski, A., Kang, M. B., Berishaj, M., Nnadi, N., Chowdury, A., Diwadkar, V. A., & Salch, A. (2023). Statistical inference for persistent homology applied to simulated fmri time series data. *Foundations of Data Science*, *5*(1), 1–25. https://doi.org/10.3934/fods.2022014

Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F., & Ziegelmeier, L. (2017). Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, *18*.

Ali, D., Asaad, A., Jimenez, M., Nanda, V., Paluzo-Hidalgo, E., & Soriano-Trigueros, M. (2023). A survey of vectorization methods in topological data analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *45*(12), 14069–14080. https://doi.org/10.1109/TPAMI.2023.3308391

Andreella, A., De Santis, R., Vesely, A., & Finos, L. (2023). Procrustes-based distances for exploring between-matrices similarity. *Statistical Methods & Applications*, *32*(3), 867–882.

Atasoy, S., Donnelly, I., & Pearson, J. (2016). Human brain networks function in connectome-specific harmonic waves. *Nature Communications*, *7*.

Axler, S. J. (1997). *Linear algebra done right*. Springer New York.

Bauer, U. (2015). *Persistent homology algorithm toolbox*. https://github.com/Ripser/ripser

Bauer, U., Kerber, M., & Reininghaus, J. (2013). *Persistent homology algorithm toolbox*. https://www.sciencedirect.com/science/article/pii/S0747717116300098

Bobadilla-Suarez, S., Ahlheim, C., Mehrotra, A., Panos, A., & Love, B. (2019). Measures of neural similarity. *Computational Brain & Behavior*, *3*. https://doi.org/10.1007/s42113-019-00068-5

Bracci, S., & Op de Beeck, H. (2016). Dissociations and associations between shape and category representations in the two visual pathways. *The Journal of Neuroscience : the Official Journal of the Society for Neuroscience*, *36*(2), 432–444. https://doi.org/10.1523/JNEUROSCI.2314-15.2016

Bubenik, P. (2015). Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, *16*(1), 77–102.

Bubenik, P., & Dlotko, P. (2017). A persistence landscapes toolbox for topological statistics [Algorithms and Software for Computational Topology]. *Journal of Symbolic Computation*, *78*, 91–114. https://doi.org/https://doi.org/10.1016/j.jsc.2016.03.009

Cai, M. B., Schuck, N. W., Pillow, J. W., & Niv, Y. (2019). Representational structure or task structure? bias in neural representational similarity analysis and a bayesian method for reducing bias. *PLOS Computational Biology*, *15*(5), 1–30. https://doi.org/10.1371/journal.pcbi.1006299

Calhoun, V., & Sui, J. (2016). Multimodal fusion of brain imaging data: A key to finding the missing link(s) in complex mental illness [Brain Connectivity in Psychopathology]. *Biological Psychiatry: Cognitive Neuroscience and Neuroimaging*, *1*(3), 230–244. https://doi.org/https://doi.org/10.1016/j.bpsc.2015.12.005

Carlsson, G. E., Ishkhanov, T., de Silva, V., & Zomorodian, A. (2007). On the local behavior of spaces of natural images. *International Journal of Computer Vision*, *76*, 1–12.

Carrière, M., Cuturi, M., & Oudot, S. (2017, August). Sliced Wasserstein kernel for persistence diagrams. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (pp. 664–673, Vol. 70). PMLR.

Charest, I., Kriegeskorte, N., & Kay, K. (2018). Glmdenoise improves multivariate pattern analysis of fmri data. *NeuroImage*, *183*. https://doi.org/10.1016/j.neuroimage.2018.08.064

Chaudhuri, R., Gerçek, B., Pandey, B., Peyrache, A., & Fiete, I. (2019). The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep. *Nature Neuroscience*, *22*(9), 1512–1520.

Chazal, F., Fasy, B. T., Lecci, F., Rinaldo, A., & Wasserman, L. (2014). Stochastic convergence of persistence landscapes and silhouettes. *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, 474–483. https://doi.org/10.1145/2582112.2582128

Chen, X., Martin, R., & Fischer-Baum, S. (2021). Challenges for using representational similarity analysis to infer cognitive processes: A demonstration from interactive activation models of word reading. *Proceedings of the Annual Conference of the Cognitive Science Society*, *43*. https://par.nsf.gov/biblio/10321541

Chung, S., & Abbott, L. (2021). Neural population geometry: An approach for understanding biological and artificial neural networks [Computational Neuroscience]. *Current Opinion in Neurobiology*, *70*, 137–144.

Churchland, P. (1986). Some reductive strategies in cognitive neurobiology. *Mind*, *95*(379), 279–309.

Churchland, P. (1995). *The engine of reason, the seat of the soul: A philosophical journey into the brain*. London. https://books.google.ca/books?id=J9D4uhcNvdIC

Cohen, M. A., Dilks, D. D., Koldewyn, K., Weigelt, S., Feather, J., Kell, A. J., Keil, B., Fischl, B., Zöllei, L., Wald, L., Saxe, R., & Kanwisher, N. (2019). Representational similarity precedes category selectivity in the developing ventral visual pathway. *NeuroImage*, *197*, 565–574. https://doi.org/10.1016/j.neuroimage.2019.05.010

Cohen-Steiner, D., Edelsbrunner, H., & Harer, J. (2007). Stability of persistence diagrams. *Discrete & Computational Geometry*, *37*(1), 103–120.

Connolly, A. C., Guntupalli, J. S., Gors, J., Hanke, M., Halchenko, Y. O., Wu, Y.-C., Abdi, H., & Haxby, J. V. (2012). The representation of biological classes in the human brain. *Journal of Neuroscience*, *32*(8), 2608–2618. https://www.jneurosci.org/content/32/8/2608

Curto, C. (2017). What can topology tell us about the neural code. *Bulletin of the American Mathematical Society*, *54*(1), 63–78.

De Marchi, S., Lot, F., Marchetti, F., & Poggiali, D. (2022). Variably scaled persistence kernels (vspks) for persistent homology applications. *Journal of Computational Mathematics and Data Science*, *4*, 100050. https://doi.org/https://doi.org/10.1016/j.jcmds.2022.100050

de Silva, V., Morozov, D., & Vejdemo-Johansson, M. (2011a). Persistent cohomology and circular coordinates. *Discrete & Computational Geometry*, *45*(4).

de Silva, V., Morozov, D., & Vejdemo-Johansson, M. (2011b). Dualities in persistent (co)homology. *Inverse Problems*, *27*(12), 124003. https://doi.org/10.1088/0266-5611/27/12/124003

Devereux, B. J., Clarke, A., Marouchos, A., & Tyler, L. K. (2013). Representational similarity analysis reveals commonalities and differences in the semantic processing of words and objects. *The Journal of Neuroscience : the Official Journal of the Society for Neuroscience*, *33*(48), 18906–18916. https://doi.org/10.1523/JNEUROSCI.3809-13.2013

Dhillon, I. S., Guan, Y., & Kulis, B. (2004). *A unified view of kernel k-means, spectral clustering and graph cuts*. Citeseer.

Diedrichsen, J., Berlot, E., Mur, M., Schütt, H., & Kriegeskorte, N. (2020). Comparing representational geometries using the unbiased distance correlation. *arXiv*.

Dujmović, M., Bowers, J. S., Adolfi, F., & Malhotra, G. (2022). The pitfalls of measuring representational similarity using representational similarity analysis. *bioRxiv*, *1*. https://doi.org/10.1101/2022.04.05.487135

Edelman, S. (1998). Representation is representation of similarities. *Behavioral and Brain Sciences*, *21*(4), 449–467. https://doi.org/10.1017/S0140525X98001253

Edelsbrunner, H., Letscher, D., & Zomorodian, A. (2000). Topological persistence and simplification. *Discrete & Computational Geometry*, *28*, 511–533.

Ellis, C., Lesnick, M., Henselman, G., Keller, B., & Cohen, J. (2019). Feasibility of topological data analysis for event-related fmri. *Network Neuroscience*, *3*, 1–12. https://doi.org/10.1162/netn_a_00095

Fasy, B., Kim, J., Lecci, F., Maria, C., Millman, D., & Rouvreau, V. (2021). *Tda: Statistical tools for topological data analysis* [R package version 1.7.7]. https://CRAN.R-project.org/package=TDA

Fasy, B., Lecci, F., Rinaldo, A., Wasserman, L., Balakrishnan, S., & Singh, A. (2014). Confidence sets for persistence diagrams. *The Annals of Statistics*, *42*, 2301–2339.

Fisher, R. A., et al. (1936). Statistical methods for research workers. *Statistical methods for research workers.*, (6th Ed).

Geniesse, C., Sporns, O., Petri, G., & Saggar, M. (2019). Generating dynamical neuroimaging spatiotemporal representations (DyNeuSR) using topological data analysis. *Network Neuroscience*, *3*(3), 763–778. https://doi.org/10.1162/netn_a_00093

Gerber, S., Tasdizen, T., Joshi, S., & Whitaker, R. (2009). On the manifold structure of the space of brain images. In G.-Z. Yang, D. Hawkes, D. Rueckert, A. Noble, & C. Taylor (Eds.), *Medical image computing and computer-assisted intervention – miccai 2009* (pp. 305–312). Springer Berlin Heidelberg.

Giusti, C., Ghrist, R., & Bassett, D. (2016). Two's company, three (or more) is a simplex: Algebraic-topological tools for understanding higher-order structure in neural data. *Journal of Computational Neuroscience*, *41*. https://doi.org/10.1007/s10827-016-0608-6

Giusti, C., Pastalkova, E., Curto, C., & Itskov, V. (2015). Clique topology reveals intrinsic geometric structure in neural correlations. *Proceedings of the National Academy of Sciences*, *112*(44), 13455–13460. https://doi.org/10.1073/pnas.1506407112

Glasser, M. F., Coalson, T. S., Robinson, E. C., Hacker, C. D., Harwell, J., Yacoub, E., Ugurbil, K., Andersson, J., Beckmann, C. F., Jenkinson, M., Smith, S. M., & Van Essen, D. C. (2016). A multi-modal parcellation of human cerebral cortex. *Nature*, *536*(7615), 171–178.

Golarai, G., Liberman, A., & Grill-Spector, K. (2017). Experience shapes the development of neural substrates of face processing in human ventral temporal cortex. *Cerebral Cortex*, *27*(2), 1229–1244. https://doi.org/10.1093/cercor/bhv314

Goodman, N. (1951). *The structure of appearance*. Harvard University Press. https://books.google.ca/books?id=F_kHAQAAIAAJ

Gracia-Tabuenca, Z., Díaz-Patiño, J. C., Arelio, I., & Alcauter, S. (2020). Topological data analysis reveals robust alterations in the whole-brain and frontal lobe functional connectomes in attention-deficit/hyperactivity disorder. *eneuro*.

Gracia-Tabuenca, Z., Díaz-Patiño, J. C., Arelio, I., Moreno, M. B., Barrios, F. A., & Alcauter, S. (2021). Development of the functional connectome topology in adolescence: Evidence from topological data analysis. *bioRxiv*. https://doi.org/10.1101/2021.10.04.463103

Haim Meirom, S., & Bobrowski, O. (2022). Unsupervised geometric and topological approaches for cross-lingual sentence representation and comparison. *Proceedings of the 7th Workshop on Representation Learning for NLP*, 173–183. https://doi.org/10.18653/v1/2022.repl4nlp-1.18

Hatcher, A. (2002). *Algebraic topology*. Cambridge University Press.

Hendriks, M. H. A., Daniels, N., Pegado, F., & Op de Beeck, H. P. (2017). The effect of spatial smoothing on representational similarity in a simple motor paradigm. *Frontiers in Neurology*, *8*, 222–222. https://doi.org/10.3389/fneur.2017.00222

Hensel, F., Moor, M., & Rieck, B. (2021). A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, *4*, 681108. https://doi.org/10.3389/frai.2021.681108

Huster, R., Yu, Q., Segall, J., & Calhoun, V. (2013). Function-structure associations of the brain: Evidence from multimodal connectivity and covariance studies. *NeuroImage*, *102*. https://doi.org/10.1016/j.neuroimage.2013.09.044

Islambekov, U., & Luchinsky, A. (2022). *Tdavec: Vector summaries of persistence diagrams* [R package version 0.1.1].

I.V, A., N, S., & .M, D. (2015). Decoding multiple subject fmri data using manifold based representation of cognitive state neural signatures. *International Journal of Computer Applications*, *115*, 1–7. https://doi.org/10.5120/20224-2512

Jo, H. J., Saad, Z. S., Simmons, W. K., Milbury, L. A., & Cox, R. W. (2010). Mapping sources of correlation in resting state fmri, with artifact detection and removal. *NeuroImage*, *52*(2), 571–582. https://doi.org/https://doi.org/10.1016/j.neuroimage.2010.04.246

Kamitani, Y., & Tong, F. (2005). Decoding the visual and subjective contents of the human brain. *Nature Neuroscience*, *8*(5), 679–685. https://doi.org/10.1038/nn1444

Kang, L., Xu, B., & Morozov, D. (2021). Evaluating state space discovery by persistent cohomology in the spatial representation system. *Frontiers in Computational Neuroscience*, *15*. https://doi.org/10.3389/fncom.2021.616748

Kanwisher, N., McDermott, J., & Chun, M. M. (1997). The fusiform face area: A module in human extrastriate cortex specialized for face perception. *Journal of Neuroscience*, *17*(11), 4302–4311.

Kerber, M., Morozov, D., & Nigmetov, A. (2017). Geometry helps to compare persistence diagrams. *ACM Journal of Experimental Algorithmics*, *22*. https://doi.org/10.1145/3064175

Kiani, R., Esteky, H., Mirpour, K., & Tanaka, K. (2007). Object category structure in response patterns of neuronal population in monkey inferior temporal cortex [PMID: 17428910]. *Journal of Neurophysiology*, *97*(6), 4296–4309. https://doi.org/10.1152/jn.00024.2007

Kriegeskorte, N., Mur, M., & Bandettini, P. (2008). Representational similarity analysis – connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, *2*, 4. https://doi.org/10.3389/neuro.06.004.2008

Kriegeskorte, N., Mur, M., Ruff, D. A., Kiani, R., Bodurka, J., Esteky, H., Tanaka, K., & Bandettini, P. A. (2008). Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron*, *60*(6), 1126–1141.

Kriegeskorte, N. (2009). Relating population-code representations between man, monkey, and computational models. *Frontiers in Neuroscience*, *3*. https://doi.org/10.3389/neuro.01.035.2009

Kriegeskorte, N., & Diedrichsen, J. (2019). Peeling the onion of brain representations [PMID: 31283895]. *Annual Review of Neuroscience*, *42*(1), 407–432. https://doi.org/10.1146/annurev-neuro-080317-061906

Kriegeskorte, N., Goebel, R., & Bandettini, P. (2006). Information-based functional brain mapping. *Proceedings of the National Academy of Sciences*, *103*(10), 3863–3868. https://doi.org/10.1073/pnas.0600244103

Kriegeskorte, N., & Kievit, R. A. (2013). Representational geometry: Integrating cognition, computation, and the brain. *Trends in Cognitive Sciences*, *17*(8), 401–412. https://doi.org/https://doi.org/10.1016/j.tics.2013.06.007

Krishnapriyan, A. S., Montoya, J., Haranczyk, M., Hummelshøj, J., & Morozov, D. (2021). Machine learning with persistent homology and chemical word embeddings improves prediction accuracy and interpretability in metal-organic frameworks. *Scientific Reports*, *11*(1), 8888.

Kusano, G., Fukumizu, K., & Hiraoka, Y. (2018). Kernel method for persistence diagrams via kernel embedding and weight factor. *Journal of Machine Learning Research*, *18*(189), 1–41. http://jmlr.org/papers/v18/17-317.html

Laakso, A. (2000). Content and cluster analysis: Assessing representational similarity in neural systems. *Philosophical Psychology*, *13*.

Lacombe, T., Montassif, H., Soriano-Trigueros, M., Spreeman, G., & Takenouchi, M. (2019). *The gudhi library is a generic open source c++ library, with a python interface, for topological data analysis (tda) and higher dimensional geometry understanding*. https://gudhi.inria.fr/

Le, T., & Yamada, M. (2018). Persistence fisher kernel: A riemannian manifold kernel for persistence diagrams. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/959ab9a0695c467e7caf75431a872e5c-Paper.pdf

Lee, H., Chung, M. K., Kang, H., Kim, B.-N., & Lee, D. S. (2011). Discriminative persistent homology of brain networks. *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 841–844. https://doi.org/10.1109/ISBI.2011.5872535

Liu, T. (2016). Noise contributions to the fmri signal: An overview. *NeuroImage*, *143*, 141–151.

Liu, Y., Liang, M., Zhou, Y., He, Y., Hao, Y., Song, M., Yu, C., Liu, H., Liu, Z., & Jiang, T. (2008). Disrupted small-world networks in schizophrenia. *Brain*, *131*(4), 945–961.

Mack, S., Kandel, E., Jessell, T., Schwartz, J., Siegelbaum, S., & Hudspeth, A. (2013). *Principles of neural science, fifth edition*. McGraw-Hill Education. https://books.google.ca/books?id=s64z-LdAIsEC

Mead, A. (1992). Review of the development of multidimensional scaling methods. *Journal of the Royal Statistical Society. Series D (The Statistician)*, *41*(1), 27–39.

Morozov, D. (2017). *Dionysus is a c++ library for computing persistent homology*. https://mrzv.org/software/dionysus2/

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT press.

Nassi, J., & Callaway, E. (2009). Parallel processing strategies of the primate visual system. *Nature Reviews Neuroscience*, *10*, 360–72. https://doi.org/10.1038/nrn2619

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pillai, A. S., & Jirsa, V. K. (2017). Symmetry breaking in space-time hierarchies shapes brain dynamics and behavior. *Neuron*, *94*(5), 1010–1026. https://doi.org/https://doi.org/10.1016/j.neuron.2017.05.013

Prince, J. S., Charest, I., Kurzawski, J. W., Pyles, J. A., Tarr, M. J., & Kay, K. N. (2022). Improving the accuracy of single-trial fmri response estimates using glmsingle (P. Kok, F. P. de Lange, P. Kok, & B. Turner, Eds.). *eLife*, *11*, e77599. https://doi.org/10.7554/eLife.77599

Reininghaus, J., Huber, S., Bauer, U., & Kwitt, R. (2015). A stable multi-scale kernel for topological machine learning. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4741–4748.

Reuter, M., Wolter, F.-E., & Peinecke, N. (2006). Laplace–beltrami spectra as 'shape-dna' of surfaces and solids [Symposium on Solid and Physical Modeling 2005]. *Computer-Aided Design*, *38*(4), 342–366. https://doi.org/https://doi.org/10.1016/j.cad.2005.10.011

Robinson, A., & Turner, K. (2017). Hypothesis testing for topological data analysis. *Journal of Applied and Computational Topology*, *1*.

Rothlein, D., & Rapp, B. (2014). The similarity structure of distributed neural responses reveals the multiple representations of letters. *NeuroImage*, *89*, 331–344. https://doi.org/https://doi.org/10.1016/j.neuroimage.2013.11.054

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations [Computational Models of the Brain]. *NeuroImage*, *52*(3), 1059–1069. https://doi.org/https://doi.org/10.1016/j.neuroimage.2009.10.003

Saggar, M., Shine, J. M., Liégeois, R., Dosenbach, N. U. F., & Fair, D. (2022). Precision dynamical mapping using topological data analysis reveals a hub-like transition state at rest. *Nature Communications*, *13*(1), 4791.

Saggar, M., Sporns, O., Gonzalez-Castillo, J., Bandettini, P. A., Carlsson, G. E., Glover, G. H., & Reiss, A. L. (2018). Towards a new approach to reveal dynamical organization of the brain using topological data analysis. *Nature Communications*, *9*.

Salch, A., Regalski, A., Abdallah, H., Suryadevara, R., Catanzaro, M., & Diwadkar, V. (2021). From mathematics to medicine: A practical primer on topological data analysis (tda) and the development of related analytic tools for the functional discovery of latent structure in fmri data. *PLoS ONE*, *16*(8).

Saul, N., & Tralie, C. (2019). Scikit-tda: Topological data analysis for python. https://doi.org/10.5281/zenodo.2533369

Scholkopf, B., Smola, A., & Muller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

Seitzman, B. A., Snyder, A. Z., Leuthardt, E. C., & Shimony, J. S. (2019). The state of resting state networks. *Topics in Magnetic Resonance Imaging : TMRI*, *28*(4), 189–196.

Shahbazi, M., Shirali, A., Aghajan, H., & Nili, H. (2021). Using distance on the riemannian manifold to compare representations in brain and in models. *NeuroImage*, *239*, 118271. https://doi.org/https://doi.org/10.1016/j.neuroimage.2021.118271

Shepard, R. N., & Chipman, S. (1970). Second-order isomorphism of internal representations: Shapes of states. *Cognitive Psychology*, *1*(1), 1–17. https://doi.org/https://doi.org/10.1016/0010-0285(70)90002-2

Shine, J. M., Breakspear, M., Bell, P. T., Ehgoetz Martens, K. A., Shine, R., Koyejo, O., Sporns, O., & Poldrack, R. A. (2019). Human cognition involves the dynamic integration of neural activity and neuromodulatory systems. *Nature Neuroscience*, *22*(2), 289–296.

Singh, G., Mémoli, F., & Carlsson, G. E. (2007). Topological methods for the analysis of high dimensional data sets and 3d object recognition. *SPBG*.

Singh, G., Mémoli, F., Ishkhanov, T., Sapiro, G., Carlsson, G., & Ringach, D. (2008). Topological analysis of population activity in visual cortex. *Journal of Vision*, *8*, 11.1–18. https://doi.org/10.1167/8.8.11

Sizemore, A. E., Giusti, C., Kahn, A., Vettel, J., Betzel, R., & Bassett, D. (2018). Cliques and cavities in the human connectome. *Journal of Computational Neuroscience*, *44*, 1–31. https://doi.org/10.1007/s10827-017-0672-6

Sizemore, A. E., Phillips-Cremins, J. E., Ghrist, R., & Bassett, D. S. (2019). The importance of the whole: Topological data analysis for the network neuroscientist. *Network Neuroscience*, *3*(3), 656–673. https://doi.org/10.1162/netn_a_00073

Somasundaram, E. V., Brown, S. E., Litzler, A., Scott, J. G., & Wadhwa, R. R. (2021). Benchmarking R packages for Calculation of Persistent Homology. *The R Journal*, *13*(1), 184–193. https://doi.org/10.32614/RJ-2021-033

Stamm, A. (2023). *Rgudhi: An interface to the gudhi library for topological data analysis* [R package version 0.2.0]. https://CRAN.R-project.org/package=rgudhi

Supekar, K., Menon, V., Rubin, D., Musen, M., & Greicius, M. D. (2008). Network analysis of intrinsic functional brain connectivity in alzheimer's disease. *PLoS computational biology*, *4*(6), e1000100.

Szekely, G., Rizzo, M., & Bakirov, N. (2008). Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, *35*. https://doi.org/10.1214/009053607000000505

Tamir, D. I., Thornton, M. A., Contreras, J. M., & Mitchell, J. P. (2016). Neural evidence that three dimensions organize mental state representation: Rationality, social impact, and valence. *Proceedings of the National Academy of Sciences*, *113*(1), 194–199. https://doi.org/10.1073/pnas.1511905112

Tauzin, G., Lupo, U., Tunstall, L., Perez, J. B., Caorsi, M., Medina-Mardones, A., Dassatti, A., & Hess, K. (2020). Giotto-tda: A topological data analysis toolkit for machine learning and data exploration.

Tenenbaum, J. B., Silva, V. d., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323. https://doi.org/10.1126/science.290.5500.2319

Turner, K. (2013). Means and medians of sets of persistence diagrams. *arXiv*.

Turner, K., Mileyko, Y., Mukherjee, S., & Harer, J. (2014). Frechet means for distributions of persistence diagrams. *Discrete & Computational Geometry*, *52*(1), 44–70.

Ungerleider, L. G., & Mishkin, M. (1982). Two cortical visual systems. In D. J. Ingle, M. A. Goodale, & R. J. W. Mansfield (Eds.), *Analysis of visual behavior* (pp. 549–586). MIT Press.

Viviani, R. (2021). Overcoming bias in representational similarity analysis. https://doi.org/10.48550/ARXIV.2102.08931

Wadhwa, R. R., Williamson, D. F. K., Dhawan, A., & Scott, J. G. (2018). Tdastats: R pipeline for computing persistent homology in topological data analysis. *Journal of Open Source Software*, *3*(28), 860. https://doi.org/10.21105/joss.00860

Wang, L., Zhu, C., He, Y., Zang, Y., Cao, Q., Zhang, H., Zhong, Q., & Wang, Y. (2009). Altered small-world brain functional networks in children with attention-deficit/hyperactivity disorder. *Human Brain Mapping*, *30*(2), 638–649.

Wasserman, E., Chakroff, A., Saxe, R., & Young, L. (2017). Illuminating the conceptual structure of the space of moral violations with searchlight representational similarity analysis. *NeuroImage*, *159*, 371–387. https://doi.org/https://doi.org/10.1016/j.neuroimage.2017.07.043

Yen, P. T.-W., & Cheong, S. A. (2021). Using topological data analysis (tda) and persistent homology to analyze the stock markets in singapore and taiwan. *Frontiers in Physics*, *9*. https://doi.org/10.3389/fphy.2021.572216

Yeo, B. T., Krienen, F., Sepulcre, J., Sabuncu, M., Lashkari, D., Hollinshead, M., Roffman, J., Smoller, J., Zollei, L., Polimeni, J., Fischl, B., Liu, H., & Buckner, R. (2011). The organization of the human cerebral cortex estimated by functional correlation. *Journal of Neurophysiology*, *106*, 1125–65. https://doi.org/10.1152/jn.00338.2011

You, K., & Park, H.-J. (2022). Geometric learning of functional brain network on the correlation manifold. *Scientific Reports*, *12*(1), 17752. https://doi.org/10.1038/s41598-022-21376-0

You, K., & Yu, B. (2021). *Tdakit: Toolkit for topological data analysis* [R package version 0.1.2]. https://CRAN.R-project.org/package=TDAkit

Yousefnezhad, M., Sawalha, J., Selvitella, A., & Zhang, D. (2021). Deep representational similarity learning for analyzing neural signatures in task-based fmri dataset. *Neuroinformatics*, *19*(3), 417–431.

Zomorodian, A. (2010). The tidy set: A minimal simplicial set for computing homology of clique complexes. *Proceedings of the Twenty-Sixth Annual Symposium on Computational Geometry*, 257–266. https://doi.org/10.1145/1810959.1811004

Zomorodian, A., & Carlsson, G. (2005). Computing persistent homology. *Discrete and Computational Geometry*, *33*, 249–274. https://doi.org/10.1007/s00454-004-1146-y