University of Sheffield

Time-Series Clustering and Visualization for Insights into Multimorbidity Progression

University of Sheffield | Healthy Lifespan Institute

Benediktas Valys

Supervisors:

Dr. Maria-Cruz Villa-Uriol

Dr. Mauricio A. Álvarez

A thesis submitted for the degree of
Doctor of Philosophy

in the

School of Computer Science

September 19, 2025

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and are my own work, result of my PhD research. This work has not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university.

Name: Benediktas Valys

Date: October 25, 2024

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Dr. Maria-Cruz Villa-Uriol and Dr. Mauricio A. Álvarez, for their invaluable guidance, support, and encouragement throughout the course of my PhD. Their expertise and mentorship have been instrumental in shaping this research, and I am immensely grateful for their patience and the countless insightful discussions.

I would like to thank the HELSI institute for their training and support throughout my research journey. The training sessions and opportunities for learning provided by HELSI have greatly contributed to my growth as a researcher.

I am also incredibly grateful to my collaborators in Sweden from the SNAC-k study, as well as the teams involved in CARE75+ and Connected Bradford. Their contributions, insights, and dedication have enriched this research in countless ways, and I truly appreciate their willingness to share their expertise and collaborate.

To my mother, thank you for being my pillar of strength and providing me with unwavering support throughout my academic journey. Your belief in me has been a source of constant motivation.

To my partner Irina, thank you for your endless love, understanding, and for always being there during the ups and downs of this journey. Your presence has made even the most challenging moments bearable.

I am deeply grateful to my sisters, who have always been my greatest cheerleaders and have provided me with so much love and encouragement. Their support has kept me grounded and driven.

Lastly, I would like to thank all my friends for providing much-needed distractions when I needed a break. Your friendship has been invaluable throughout this journey, and I am so grateful to have you all by my side.

This thesis would not have been possible without the love, support, and guidance of all these amazing individuals. Thank you.

Abstract

The increasing availability of vast amounts of data in electronic health records (EHR) offers immense opportunities to extract valuable insights, particularly through the application of machine learning techniques like clustering. This thesis focuses on clustering time-series data extracted from medical records, with the aim of identifying meaningful clusters of patient sequences. While clustering methods are well-established for static datasets, clustering time-series data presents unique challenges, especially when it comes to selecting the most relevant solution from many valid clustering outputs.

In this work, we develop a two-stage methodology for clustering time-series data. The first stage simplifies high-dimensional sequence data, while the second stage focuses on identifying clusters within these sequences. We also address the issue of comparing multiple clustering solutions by introducing a novel approach that combines a graphical user interface (GUI) with a graph-based representation of the relationships between different clustering solutions. This framework allows for intuitive, simultaneous exploration and comparison of multiple valid solutions, helping to reduce the space of possible results and aiding in the interpretation of alternative outcomes.

Our methodology is applied to the domain of multimorbidity, a significant healthcare challenge characterised by the coexistence of multiple chronic conditions. By applying our tools to multimorbidity datasets, we gain insights into the progression of chronic illnesses and their interactions.

Contents

List of Figures

List of Tables

Notation

Symbols and notation

The notation provided below is used throughout this thesis to represent various aspects of multivariate time series datasets and their matrix representations.

$\mathbf{X}$	Multivariate time series.
$\mathbf{x}_t$	Observation at time t, a binary vector of length D.
D	Number of possible events in $\mathbf{X}$.
$x_{d,t}$	Value of the d-th variable at time t.
T	Total number of observations in $\mathbf{X}$.
$\mathcal{X}$	Dataset containing multiple multivariate time series.
$\mathbf{X}_i$	i-th time series in the dataset.
N	Total number of time series in $\mathcal{X}$.
$\mathcal{X}_{\text{red}}$	Multivariate time series data of reduced dimensionality.
m	Size of lower dimensional dataset $\mathcal{X}_{\text{red}}$.
$S = (e_1, e_2, \ldots, e_T)$	Redefined multivariate time series using timepoint cluster labels.
Q	Number of HMM states.
$\mathbf{C} = \{C_1, C_2, \ldots, C_v\}$	Sequence clustering solution.
$C_v = \{S_1, S_2, \ldots, S_{\lvert C_v \rvert}\}$	Sequence cluster.
$E_k = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{\lvert E_k \rvert}\}$	Timepoint cluster or Event Group k.
EGS_D^k	Event group score for event D in timepoint cluster k.
$w = \alpha c_{\text{sil}} + (1-\alpha) s_{\text{sil}}$	Loss value optimized during hyperparameter optimization.
r	Number of iterations in hyperparameter optimisation.
c_{sil}	Silhouette index for timepoint clusters.
s_{sil}	Silhouette index for sequence clusters.
α	Mixing coefficient, representing the weight assigned to either the timepoint or sequence silhouette index.
$G(C, E)$	Graph of clustering solutions, nodes are C clustering solutions, E edges, distances between solutions.
θ	Similarity threshold for defining edges in the graph.
A	Adjacency matrix of the graph, where A_{ij} represents the weight of the edge between nodes i and j.
$\mathbf{F} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_u\}$	Set of clustering solutions.
$S(\mathbf{C}_i, \mathbf{C}_j)$	Similarity function quantifying the similarity between clustering solutions.

$G = (V, E)$	Graph constructed from clustering solutions.
$v_i \in V$	Node in the graph corresponding to a clustering solution.
$e_{ij} \in E$	Edge in the graph between nodes v_i and v_j with weight w_{ij}.
$w_{ij} = S(\mathbf{C}_i, \mathbf{C}_j)$	Weight of the edge between nodes v_i and v_j.
$G' = (V, E')$	Subgraph formed by retaining edges with weights $w_{ij} \geq \theta$.
$E' = \{e_{ij} \mid w_{ij} \geq \theta\}$	Edge set of the thresholded graph G'.
$\mathcal{C}_i$	Cohort of solutions in G'.
$s_{\text{sil}}(\mathbf{C}_j, \mathcal{C}_i)$	Silhouette index of solution $\mathbf{C}_j$ within cohort $\mathcal{C}_i$.
$P_i = \arg\max_{\mathbf{C}_j \in \mathcal{C}_i} s_{\text{sil}}(\mathbf{C}_j, \mathcal{C}_i)$	Prototype solution for cohort $\mathcal{C}_i$.

Clustering Validation Indices

The table below details various clustering quality measures used in the thesis.

Table 1: Common clustering validation indices: definition, type, and interpretation.

Index	Type	Scale (direction)	What it measures / Notes
Adjusted Rand Index (ARI)	External	$[-1, 1]$ (higher better)	Chance-adjusted pairwise agreement between predicted and ground-truth labels; 1 = perfect match, ≈ 0 = chance-level, negative = worse than chance. Invariant to label permutations; sensitive to pair-count imbalance.
Normalised Mutual Information (NMI)	External	$[0, 1]$ (higher better)	Mutual information between labelings, normalised (e.g., by mean or max entropy). Invariant to label permutations; compares solutions with different k; can be optimistic with many small clusters.
Fowlkes–Mallows Index (FMI)	External	$[0, 1]$ (higher better)	Geometric mean of precision and recall of pair co-assignment. Simple and interpretable; does not correct for chance; can be influenced by class imbalance.
Silhouette index	Internal	$[-1, 1]$ (higher better)	Mean over points of $(b - a)/\max(a, b)$ where a is the average intra-cluster dissimilarity and b is the lowest average dissimilarity to any other cluster. Higher indicates compact, well-separated clusters; it depends on the chosen distance; less reliable for highly non-convex shapes.
Davies–Bouldin (DB) index	Internal	$[0, \infty)$ (lower better)	Average, over clusters, of the ratio (within-cluster scatter)/(between-centroid separation) with the most similar other cluster. Lower indicates compact, well-separated clusters; it depends on the metric and cluster shape.

Software Dependencies

This section lists the software used for model development and analysis. The full pipeline and reproducible notebooks are available in the online appendix at GitHub (Thesis_code).

- `numpy`, `pandas`: array and DataFrame operations; data manipulation.
- `scikit-learn` (`KMeans`, `DBSCAN`, `silhouette_score`, `davies_bouldin_score`): clustering and internal validation metrics.

- `prince`: Multiple Correspondence Analysis (MCA) for dimensionality reduction of categorical data.
- `hmmlearn`, `pomegranate`: Hidden Markov Model training and probabilistic modelling for sequence analysis.
- `scipy` (`spatial.distance.squareform`): distance transformations and utilities.
- `hyperopt` (`tpe`, `hp`, `fmin`, `Trials`): hyperparameter optimisation.
- `networkx`: similarity-graph construction, connected components, layouts.
- `matplotlib`: static plots (graphs, histograms, density plots) and figure export.
- `seaborn`: statistical visualisation for distributions and demographics.
- `plotly` (`graph_objects`, `express`; `kaleido` for export): Sankey diagrams and interactive figures.
- `ipywidgets`: sliders and controls for thresholds, filters, and selections.
- `IPython.display` (`display`, `clear_output`, `HTML`): inline outputs in notebooks.
- `tqdm.notebook`: progress bars for batch computations and exports.
- `distinctipy`: distinct colour palettes for components and Event Groups.
- `dataframe_image`: rendering DataFrame summaries and metadata as images.
- `Pillow` (`PIL`): image composition and saving combined figures.

`os`, `pathlib`, `shutil`, `fnmatch`, `random`, `time`, `math`, `collections`, `itertools`.

Chapter 1

Introduction

The rapid expansion of electronic health records (EHR) offers unprecedented opportunities to advance healthcare through the analysis of real-world clinical data. These datasets contain rich information about diagnoses, treatments, comorbidities, and patient outcomes. However, this information is often complex, high-dimensional, and temporally structured, requiring advanced analytical tools to extract meaningful insights. Among such tools, clustering methods are widely used to detect patterns, identify subgroups of patients, and understand disease trajectories. Clustering has become an essential part of healthcare analytics, with applications including disease phenotyping, treatment stratification, and multimorbidity analysis [1], [2], [3].

A particularly important application of clustering in healthcare is the study of multimorbidity—the co-occurrence of multiple chronic conditions in a single individual. Multimorbidity is commonly defined as the presence of two or more chronic diseases [4], and is becoming increasingly prevalent as populations age and chronic conditions become more widespread. Understanding how multimorbidity develops and progresses over time remains a significant public health challenge. A clearer view of these trajectories can support healthcare systems in planning timely interventions, anticipating care demands, and allocating resources more efficiently [5, 6].

This thesis focuses on clustering time-series representations of patient histories, with a specific emphasis on multimorbidity progression. Patient histories can be viewed as sequences or time series of diagnostic events across several observation periods. In this thesis, the term time series is used to refer to sequences of patient-level health observations ordered over time. Specifically, we focus on categorical event sequences—diagnosis groups recorded at discrete, regular intervals—rather than continuous-valued time series. While the broader literature on time-series analysis includes methods for regularly sampled numerical data, our focus is on sequence-based representations, where the timing is implicit and structured by fixed observation windows. Therefore, our methods and analyses are more accurately aligned with sequence clustering, although we use the term "time series" at times for consistency with broader terminology in healthcare analytics. Unlike static data clustering, clustering time-series data poses additional challenges: time dependencies, temporal alignment, variable sequence lengths, and high dimensionality all complicate traditional approaches. Moreover, clustering algorithms can produce many valid solutions depending on hyperparameters and initialisation—making it difficult to interpret results and choose a "best" clustering.

The core motivation behind this work is to develop a robust, interpretable methodology for identifying meaningful patient clusters over time, particularly in multimorbid populations. Clustering these time-ordered events can help uncover important trends in patient care, such as common patterns of disease progression or early indicators of high-risk health trajectories. This is particularly useful in identifying subgroups for preventive care or targeted interventions.

However, there are several challenges in applying clustering to longitudinal health data:

1. **High dimensionality**: Patient time series often contain dozens or hundreds of variables (diagnoses, treatments), many of which are sparse.
2. **Multiple valid clusterings**: Due to algorithmic sensitivity and different similarity measures, there is often no unique best clustering.
3. **Interpretability**: Clinical stakeholders need results that are both statistically robust and clinically meaningful.
4. **Irregularities**: Healthcare data can be noisy, incomplete, or unevenly sampled in time, which complicates model assumptions.

To address these challenges, this thesis introduces a two-stage clustering methodology for time-series data. The method involves:

1. Dimensionality reduction and clustering of individual timepoints using a technique like Multiple Correspondence Analysis (MCA) to form "event groups" that reduce the complexity of raw input.
2. Clustering full sequences of these timepoint groupings to identify trajectories of multimorbidity progression among patients.

In addition, a novel graph-based framework for comparing multiple clustering solutions is presented, alongside an interactive GUI (ClusterView) for visual exploration. This tool is designed to help analysts and clinical researchers evaluate clustering solutions not only quantitatively, but visually and comparatively. This tool is application-independent and is valid for any area of clustering analysis.

The methodology is validated on two large, real-world datasets: the SNAC-K study from Sweden and the CARE75+ study from the UK. These datasets represent older populations with diverse multimorbidity patterns and longitudinal follow-up.

Taken together, the methods developed in this thesis aim to support the exploration and interpretation of complex, temporally structured health data—particularly in the context of multimorbidity progression. By combining a sequence-based clustering framework with tools for comparing and visualising alternative solutions, this work offers a scalable and interpretable approach for uncovering meaningful patterns in patient trajectories. These contributions enable analysts and clinicians to identify common pathways through which chronic conditions co-occur and evolve, improving the capacity to extract actionable insights from longitudinal datasets. Ultimately, this thesis provides both methodological and practical advancements that strengthen the use of clustering in real-world healthcare applications and contribute to a deeper understanding of how disease patterns unfold over time.

Chapter 2

Literature review

2.1 Cluster Analysis

Clustering is a fundamental unsupervised machine learning technique used to group data points or objects so that those within the same group (cluster) are more similar to each other than to those in different groups [7, 8]. The primary aim is to uncover inherent patterns or structures in the data without predefined labels. Similarity between data points can be defined in various ways, depending on the nature of the data and the intended application. Different clustering algorithms use this idea in distinct ways, and not all methods optimise a single global objective function. For example, k-means clustering explicitly minimises the within-cluster sum of squared distances to cluster centroids, while hierarchical clustering builds a nested sequence of partitions based on a linkage criterion without optimising a single overarching objective.

Clustering algorithms can be broadly categorised into several types. In **partitional clustering**, such as k-means [9] or Gaussian Mixture Models (GMMs) [10], the dataset is divided into a fixed number of non-overlapping clusters, often guided by an optimisation criterion. In **hierarchical clustering** [11], clusters are formed in a nested manner, either through successive merging of smaller clusters (agglomerative) or successive splitting of larger clusters (divisive). In **density-based clustering**, such as DBSCAN [12], clusters are defined as regions of high point density separated by areas of lower density, enabling the discovery of arbitrarily shaped clusters and noise points. **Model-based clustering** approaches, including the Expectation–Maximisation (EM) algorithm applied to GMMs, assume that the data are generated from a mixture of underlying probability distributions, with each cluster corresponding to one distribution component.

Clustering can also be categorised based on whether membership is **hard** or **soft**. In **hard clustering**, each data point belongs to exactly one cluster, as is the case for standard k-means or agglomerative hierarchical clustering. In **soft** (or **fuzzy**) clustering, points can belong to multiple clusters with varying degrees of membership, as in Fuzzy C-Means [13]. Soft clustering is particularly useful when cluster boundaries are not well-defined or when overlapping group structures are expected.

Given the diversity of clustering techniques, the choice of method should be informed by the characteristics of the dataset, the definition of similarity relevant to the problem, and the interpretability needs of the analysis. The following sections provide more detail on the

main categories of clustering methods.

2.1.1 Distance Measures

Distance and similarity measures are fundamental components of clustering algorithms, as they quantify the degree of resemblance or dissimilarity between data points [14]. The choice of measure directly influences the shape, size, and separation of clusters, and is often as important as the choice of the clustering algorithm itself. Inappropriate selection can lead to misleading results, particularly in high-dimensional or noisy datasets [15]. A *distance measure* (or metric) assigns a non-negative value to represent how far apart two objects are, satisfying properties such as non-negativity, symmetry, and the triangle inequality [16]. A *similarity measure*, in contrast, assigns higher values to pairs of objects that are more alike. Many similarity measures can be transformed into distances, and vice versa, through appropriate scaling or complement operations.

Commonly used distance measures include:

- **Euclidean distance**: the straight-line distance between two points in a multidimensional space. It is computationally efficient ($O(n)$ for vectors of length n) but can be sensitive to scaling and irrelevant features.
- **Manhattan distance** (L1 norm): the sum of absolute differences across dimensions. It can be more robust to outliers in certain contexts.
- **Cosine distance**: derived from cosine similarity, it measures the angle between two vectors and is particularly useful when the magnitude of vectors is less important than their orientation, such as in text analysis.
- **Jaccard distance**: used for binary or set-valued data, defined as one minus the ratio of the intersection size to the union size of two sets.

For time-series data, *elastic* measures such as Dynamic Time Warping (DTW) [17] and Longest Common Subsequence (LCSS) [18] allow for non-linear alignment of sequences, accommodating phase shifts and local time distortions. Model-based measures, such as those derived from autoregressive models or hidden Markov models, represent each sequence through fitted model parameters and compare these lower-dimensional representations [19].
The appropriateness of a distance measure depends on the nature of the data, the scale of features, and the clustering objectives. In high-dimensional spaces, the phenomenon of *distance concentration* can make all points appear nearly equidistant under certain metrics, necessitating either dimensionality reduction or the use of alternative similarity measures [15]. Careful consideration of these factors is critical to achieving meaningful and interpretable clustering results.

2.1.2 Partition Clustering

Partition clustering, also known as partitional clustering, is a fundamental approach to clustering in which the dataset is partitioned into a set of disjoint clusters, with each data point belonging to exactly one cluster. Unlike hierarchical clustering, which produces a hierarchical decomposition of the data, partition clustering directly assigns each data point to a specific cluster. One of the most popular algorithms for partition clustering is the k-means algorithm.

2.1.2.1 K-means Algorithm

The k-means algorithm is a widely used partition clustering technique that aims to partition n observations into k clusters, where each observation belongs to the cluster with the nearest mean. It follows these steps:

1. **Initialisation :** Choose k initial cluster centroids randomly from the data points.
2. **Assignment:** Assign each data point to the nearest centroid, forming k clusters.
3. **Update Centroids:** Recalculate the centroid of each cluster based on the mean of the data points assigned to it.
4. **Repeat:** Repeat steps 2 and 3 until convergence, i.e., until the centroids no longer change significantly or until a maximum number of iterations is reached.

The algorithm aims to minimise the within-cluster sum of squares, which measures the compactness of the clusters. However, k-means is sensitive to the initial choice of centroids and may converge to a local minimum, leading to different clustering results for different initialisations. Partition clustering techniques like k-means are computationally efficient and can handle large datasets, making them suitable for a wide range of applications. However, they require the number of clusters (k) to be specified in advance and may not perform well with non-linear or non-convex clusters.

2.1.3 Hierarchical Clustering

Hierarchical clustering is a family of clustering techniques that builds a hierarchy of nested clusters without requiring the number of clusters to be specified in advance [7, 20]. The result is often visualised as a *dendrogram*, a tree-like diagram that shows the sequence of merges (in agglomerative clustering) or splits (in divisive clustering) and the distances at which these changes occur. The dendrogram enables the exploration of the data structure at multiple resolutions: by cutting the tree at different levels, one can obtain different clusterings of the same dataset. This property makes hierarchical clustering particularly useful in exploratory data analysis when the number of natural groupings is not known a priori.
Two main strategies exist: *agglomerative* (bottom-up) and *divisive* (top-down) clustering. In *agglomerative clustering*, each observation begins in its own singleton cluster, and pairs of clusters are iteratively merged according to a chosen proximity measure until a single cluster remains or a stopping criterion is met. This approach is the most common due to its conceptual simplicity and straightforward implementation. In *divisive clustering*, all points start in a single cluster, which is then recursively split into smaller clusters until each observation stands alone or a threshold is reached. Divisive approaches can reveal high-level structure before refining to smaller groups, but they are generally less computationally efficient for large datasets.

The proximity between clusters can be calculated using various *linkage criteria*:

- **Single linkage** (nearest neighbour): the distance between two clusters is defined as the shortest distance between any pair of points in the two clusters [21]. This method can produce "chained" clusters, especially in noisy datasets.

- **Complete linkage** (farthest neighbour): the distance is the largest distance between any pair of points in the two clusters [22]. This produces more compact, spherical clusters.

- **Average linkage**: the distance is the mean of all pairwise distances between points in the two clusters. It balances the chaining effect of single linkage with the compactness of complete linkage.

- **Centroid linkage**: the distance is the Euclidean distance between the centroids (mean vectors) of the two clusters.

The choice of linkage criterion affects the shape and separation of the resulting clusters, and is therefore an important methodological decision. In high-dimensional spaces, distance concentration effects can also influence cluster formation, making the choice of distance metric equally important.

In terms of computational complexity, agglomerative methods typically run in $O(n^2)$ time for n objects because they require computing and updating a full pairwise distance matrix [23]. Divisive methods can be more computationally demanding in theory, sometimes approaching $O(2^n)$ in the worst case, although practical algorithms employ heuristics or approximation strategies to make them feasible on real-world datasets.

Hierarchical clustering has the advantage of producing a complete multi-scale representation of the data without assuming a fixed number of clusters, and the dendrogram allows for flexible post-hoc decisions about the desired granularity of the clustering solution. However, the method can be sensitive to noise and outliers, and early incorrect merges or splits cannot be undone, making it less robust in some cases compared to partitional or density-based approaches. As such, it is often used in combination with other methods for validation or as an exploratory step in the analysis pipeline.

2.1.4 Fuzzy Clustering

Fuzzy clustering, also known as soft clustering, allows for the possibility of overlap between clusters, providing a more nuanced approach to data partitioning [7]. One of the most prominent algorithms in fuzzy clustering is the c-means algorithm, originally introduced by Dunn [24] and later refined by Bezdek [25]. Unlike traditional crisp clustering methods like k-means, c-means assigns a degree of membership to each object, indicating the extent to which it belongs to each cluster. The key steps of the c-means algorithm are as follows:

1. **Choose the Number of Clusters (k):** Determine the desired number of clusters to be formed.

2. **Initialise Membership Coefficients:** Assign initial fuzzy membership coefficients randomly to each object in the dataset.

3. **Compute Cluster Centroids:** Calculate the centroids of the clusters based on the fuzzy membership coefficients.

4. **Update Membership Coefficients:** Update the fuzzy membership coefficients based on the new cluster centroids.

5. **Check Convergence:** Verify whether the variance of the fuzzy coefficients falls below a predefined sensitivity threshold.

6. **Iterate Until Convergence:** If the convergence criterion is not met, repeat steps 3, 4, and 5 until convergence.

However, like many clustering algorithms, c-means is sensitive to noise and outliers, and its performance may be affected by the need for prior knowledge of the number of clusters [24].

2.1.5 Clustering Evaluation

Evaluating clustering quality is inherently challenging, as there are no labels to directly assess clustering accuracy in an unsupervised setting. Different algorithms, combined with different similarity measures, can produce a wide range of alternative clustering solutions. The quality of the final clustering depends not only on the dataset but also on the subjective interpretation of the specific problem.

To provide a systematic assessment, clustering evaluation metrics are typically divided into two categories: *internal measures* and *external measures*. Internal measures assess the quality of a clustering using only the data and the resulting assignments, quantifying aspects such as compactness within clusters and separation between clusters. External measures, in contrast, compare the clustering output to an external set of ground truth labels. The following subsections describe these two categories in more detail.

2.1.5.1 Internal Measures

Internal validation measures, such as the Silhouette index [26], Davies-Bouldin index [27], and Calinski-Harabasz score [28], assess clustering quality solely from the data and assignments, without requiring ground truth. These indices evaluate properties such as intra-cluster similarity and inter-cluster dissimilarity. Because they are functions of both the dataset and the clustering result, they can be used to compare results from different algorithms applied to the same dataset with the same feature representation and similarity metric. However, if algorithms use different data transformations, distance metrics, or operate in non-comparable feature spaces, the scores may not be directly comparable.

2.1.5.2 External Measures

External measures, on the other hand, rely on predefined ground truth labels to assess clustering performance. These metrics compare the clustering output to the known labels to determine how well the clustering algorithm has performed. Common external measures include Adjusted Rand Index (ARI), Normalised Mutual Information (NMI), and Fowlkes-Mallows Index (FMI), each providing a different perspective on the clustering quality by evaluating different aspects of the clustering result against the ground truth. By applying these quality metrics, we can objectively compare different clustering approaches and identify the most appropriate solution for a given dataset. This thorough evaluation ensures that the selected clustering algorithm not only produces distinct clusters but also aligns well with the inherent structure of the data, ultimately leading to more reliable and insightful conclusions.

2.1.6 Clustering Ensembles

Consensus clustering, also known as clustering ensembles or cluster aggregation, combines multiple individual clustering results into a single consensus solution, thereby enhancing the robustness and stability of the final clustering outcome. This approach addresses the limitations of individual clustering methods, such as sensitivity to initial settings and difficulty in determining the number of clusters (k) [29]. By aggregating results from multiple runs of a clustering algorithm with different initial conditions or parameters, consensus clustering compensates for errors in individual solutions. The process typically involves subsampling the dataset multiple times, running a chosen clustering algorithm on each subsample, and calculating pairwise consensus values stored in a symmetrical consensus matrix, which can be visualised in heatmaps to assess cluster membership stability and determine the appropriate number of clusters [30]. Various methods have been proposed for clustering ensembles, including meta-clustering, cluster merging, and cluster-based ensemble selection. Meta-clustering clusters the data multiple times with different algorithms and combines the results into a single solution, while cluster merging consolidates the solutions from different algorithms, and cluster-based ensemble selection picks the best solutions based on their similarity to others in the ensemble.

Hypergraph partitioning is a technique that has recently been proposed for clustering ensembles. A hypergraph is a generalisation of a graph where an edge can connect any number of vertices. Hypergraph partitioning aims to partition a hypergraph into several disjoint clusters based on its connectivity structure. In the context of clustering ensembles, each clustering solution can be represented as a hypergraph, where each vertex represents a data point and each hyperedge represents a cluster. Hypergraph partitioning can be used to find a consensus clustering by partitioning the ensemble hypergraph into a set of disjoint clusters [31].

2.1.7 Visualising Clustering Solutions

Visualising clustering solutions is a crucial step in understanding the structure and validity of the clusters obtained from different algorithms. Effective visualisation helps in interpreting the results and validating the performance of the clustering method. Common techniques include scatter plots, heatmaps, and dimensionality reduction methods such as PCA and t-SNE. These methods enable the display of high-dimensional data in a two- or three-dimensional space, making it easier to comprehend the clustering patterns and the relationships between different data points.

Alternative clustering visualisations can provide additional insights into the data structure. Examples include silhouette plots, which assess cluster cohesion and separation, and cluster heatmaps, which visualise the cluster centres and data distribution within clusters. These visualisations help in determining the quality and distinctiveness of the clusters, ensuring that the chosen clustering method is effective.

This section introduces the concept of Cluster Sankeys and demonstrates their utility in depicting the flow of data points across different clusters. By leveraging these visual tools, we can gain deeper insights into the structure and dynamics of our data, facilitating more

informed decisions in clustering analysis.

For algorithms such as k-means, where the number of clusters k must be specified, a common heuristic is the **elbow method**. In this procedure, k-means is run for a range of candidate values (e.g., $k = 1, \ldots, K$). For each k, the within-cluster sum of squares (WCSS)—the total intra-cluster variance—is computed. Plotting WCSS against k typically yields a monotonically decreasing curve that drops steeply before beginning to level off. The **elbow** is the point at which the marginal gain from increasing k diminishes significantly; selecting k at or near this kink balances model complexity against compactness.

In the example shown in Figure 2.1, a dataset was generated containing three isotropic Gaussian blobs. We applied k-means over multiple values of k, computed the WCSS for each fit, and plotted the results in Figure 2.2.

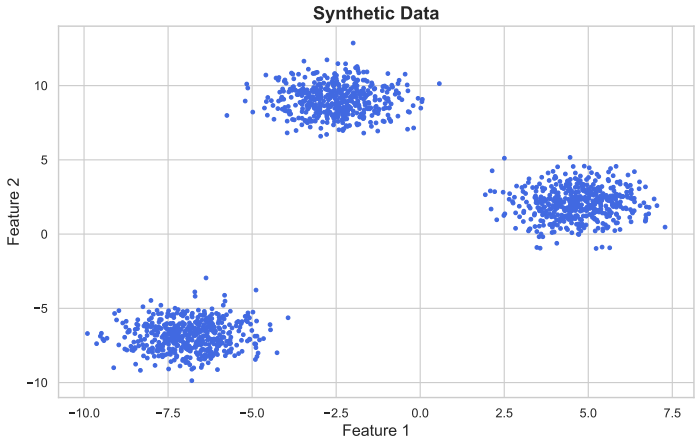

Figure 2.1: An example of a dataset comprised of 3 clouds of points.

The elbow is visible at $k = 3$, which we then selected for the final clustering visualisation shown in Figure 2.3.

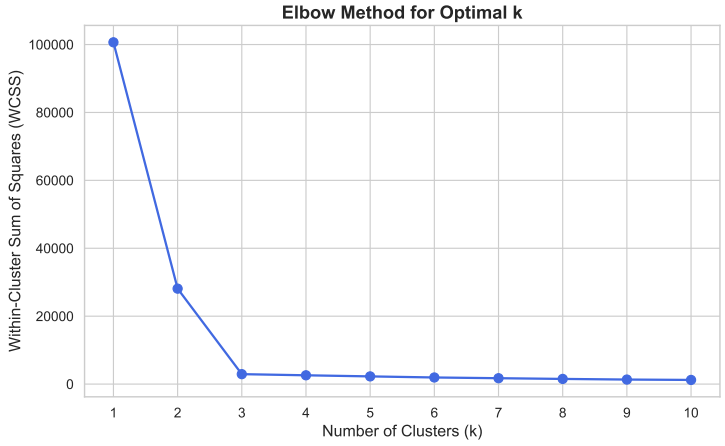

Figure 2.2: Elbow plot for the cloud of points.

This workflow demonstrates the correct sequence: obtain clustering solutions across candidate k, derive the elbow from the WCSS curve, and finally visualise the chosen partition to verify that it aligns with the data's structure.

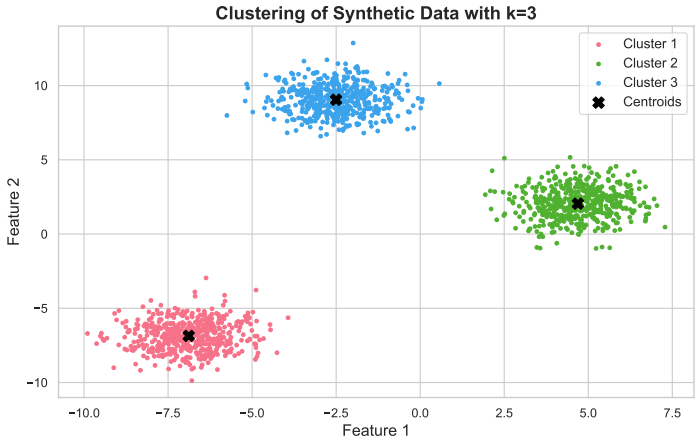

Figure 2.3: Synthetic data clustering results.

A dendrogram is a tree-like diagram that records the sequences of merges or splits in hierarchical clustering. It helps in understanding the arrangement of the clusters and the distances between them. Dendrograms are particularly useful for hierarchical clustering methods, as they provide a clear visualisation of the nested clusters. This visualisation technique is essential for interpreting the hierarchical relationships between data points and for identifying the most significant clusters. An example is presented in Figure 2.4, in the plot there are 6 data samples clustered using agglomerative hierarchical clustering. At the bottom of

the plot, all samples are in distinct clusters. Moving up the plot, the clusters are merged, which is visualised via the dendrogram.

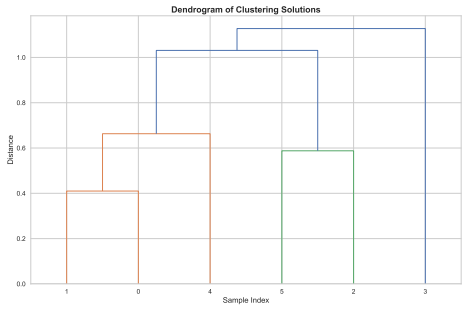

Figure 2.4: Dendrogram of clustering results.

2.2 Time series Analysis

Time series analysis deals with the study of data points collected, observed, or recorded sequentially over time. Time series data exhibits temporal dependence, where the value of a variable at a given time point is influenced by its past values. This field encompasses various statistical techniques and models to understand, interpret, and forecast time-dependent data patterns. Time series data typically comprises a sequence of observations indexed by time intervals. These observations can represent various phenomena, such as stock prices, temperature readings, sales figures, or sensor measurements, recorded at regular or irregular intervals. Time series data can exhibit trends, seasonality, cyclical patterns, and irregular fluctuations, making it essential to apply specialised analysis techniques to extract meaningful insights.

2.2.1 Modelling Time series Data

Various methods are employed to model time series data, each catering to different aspects of the temporal dependencies and patterns inherent in the data. Hidden Markov Models (HMMs) are a prominent approach used for modelling time series data, particularly in scenarios where the underlying system can be represented as a sequence of latent states [32]. HMMs are probabilistic models that capture both observed data and unobserved states, allowing for the representation of complex temporal dynamics [33]. By estimating transition probabilities between states and emission probabilities for observed data, HMMs can effectively model sequential data with hidden underlying structures. Additionally, autoregressive integrated moving average (ARIMA) models are widely used for modelling stationary time series data, capturing trends and seasonality through autoregressive and moving average components [34]. Machine learning algorithms, such as recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks, offer powerful capabilities for capturing long-range dependencies and temporal patterns in time series data [35, 36]. These diverse methods provide a toolkit for analysts and researchers to effectively model and extract insights from time series datasets across various domains.

2.2.2 Hidden Markov Models

Hidden Markov Models (HMMs) are probabilistic models widely used in various fields, including speech recognition [37], bioinformatics [38], financial market analysis [39], robotics [**thrun2005probabilistic**], and natural language processing [40]. The HMM assumes an underlying Markov process with hidden states, where each state generates an observable outcome based on a probability distribution. Formally, let $Q = \{q_1, q_2, ..., q_N\}$ be the set of hidden states, $O = \{o_1, o_2, ..., o_M\}$ be the set of possible observations, A be the state transition probability matrix with a_{ij} representing the probability of transitioning from state i to state j, and B be the observation probability matrix with $b_j(o_t)$ denoting the probability of observing o_t given the system is in state j. The initial state probabilities are represented by $\pi = \{\pi_1, \pi_2, ..., \pi_N\}$. The probability of a particular sequence of observations $O = \{o_1, o_2, ..., o_T\}$ given the model parameters $\lambda = (A, B, \pi)$ is computed using the Forward algorithm:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i),$$

where $\alpha_t(i)$ is the forward variable representing the probability of being in state i after observing the sequence up to time t. While the Forward algorithm evaluates likelihoods, the **Forward–Backward algorithm** extends this by computing posterior state probabilities. It combines forward probabilities $\alpha_t(i)$ with backward probabilities $\beta_t(i)$, where $\beta_t(i)$ denotes the probability of the partial observation sequence from $t+1$ to T, given the model is in state i at time t. The posterior probability of being in state i at time t is then expressed as:

$$P(q_t = i|O, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)}.$$

This enables estimation of the probability distribution over hidden states at each time step, which is critical for tasks such as parameter re-estimation in the Baum–Welch algorithm. The **Viterbi algorithm**, in contrast, identifies the single most likely sequence of hidden states (the Viterbi path) that explains the observed sequence. It uses dynamic programming to recursively compute:

$$\delta_t(i) = \max_{q_1,\ldots,q_{t-1}} P(q_1, \ldots, q_{t-1}, q_t = i, o_1, \ldots, o_t|\lambda),$$

where $\delta_t(i)$ represents the highest probability along a single path ending in state i at time t. By backtracking from the final state with maximum probability, the Viterbi algorithm yields the optimal state sequence. Together, the Forward, Forward–Backward, and Viterbi algorithms provide complementary tools: evaluating sequence likelihoods, inferring state posteriors, and decoding most likely state paths, respectively [37, 41]. A diagram which visualises the latent state transitions and emitted observations is presented in Figure 2.5.

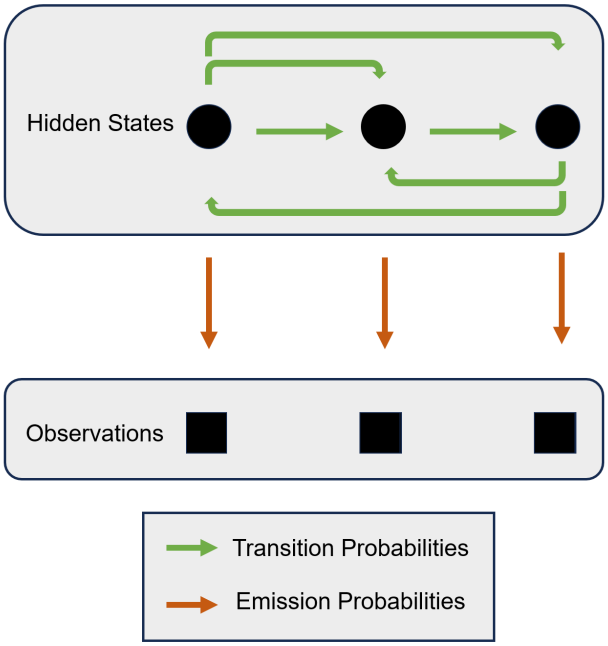

Figure 2.5: Diagram illustrating a Hidden Markov Model.

2.2.3 Time series Clustering

Given a dataset of time-series data $D = \{T_1, T_2, ..., T_N\}$, where T_n is a time series defined as a sequence of time events x_t of variable length, D can be partitioned into clusters $C = \{C_1, C_2, ..C_K\}$, where C_k represents the cluster k. This partitioning process is called **time-series clustering** and it requires the definition of a similarity measure, which will attempt to group together homogeneous time-series.

There are three families of approaches to time-series clustering:

1. **Whole time-series clustering**: treating each time-series as a separate object and assigning them all to distinct clusters
2. **Subsequence clustering**: extracting segments of a time-series and clustering these subsequences.
3. **Time point clustering**: time-series clustering based on a combination of time point values as well as their proximity in time.

This section will only focus on whole time series clustering (other families are seldom used in EHR data analysis), which can be further categorised into:

1. **Shape-based** methods can be described as raw-data clustering since they involve matching 2 time-series by stretching or contracting along the time axes.

2. **Feature-based** methods involve converting the time-series into a lower-dimensional vector which can then be compared using Euclidean distance and clustered using conventional clustering algorithms [42].

3. **Model-based** clustering involves converting raw time series data into model parameters, which are then compared and clustered [43].

In this section, the existing works related to clustering of time-series data are concentrated and discussed. Some of them are using raw time-series and some try to use reduction methods before clustering of time-series data. Generally, clustering can be broadly classified into six groups: Partitioning, Hierarchical, Grid-based, Model-based, Density-based clustering and Multi-step clustering algorithms.

Similarity measures play a pivotal role in clustering time-series data. This aspect of clustering is notably challenging due to the inherent variability in time-series, which can differ not only in length but also in sampling intervals. For instance, two time-series of varying lengths may exhibit similar shapes at different points along the time axis, making it crucial to employ similarity measures that can account for such temporal variations. The choice of an appropriate similarity measure depends on several factors, including the type of dimensionality reduction applied to the time-series data.

Traditional distance measures such as Euclidean distance are widely used for comparing fixed-length static vectors and can be computed in $O(n)$ time for two series of length n. However, in the context of time-series analysis, Euclidean distance is sensitive to misalignments in time and may fail to recognise similar patterns that are phase-shifted or occur at different speeds [44]. To address this, elastic measures such as Dynamic Time Warping (DTW) [17] and Longest Common Subsequence (LCSS) [18] allow for non-linear alignments, enabling the detection of similarities even when patterns are out of phase.

An alternative approach is model-based similarity, where each time series is represented via parameters from a fitted model (e.g., autoregressive models, hidden Markov models), and similarity is computed between these parameter sets [19]. Model-based methods can be advantageous for long sequences as they compress temporal patterns into lower-dimensional representations, reducing comparison costs.

The choice between these approaches depends on the dataset characteristics and task requirements. Shape-based measures are often effective for short, well-aligned sequences, while elastic and model-based methods are more robust to temporal distortions and varying sequence lengths [19, 44].

Type	Definition	Example Methods
Whole Time Series Clustering	Clustering of individual time series based on their similarity	k-Means, k-Medoids, Hierarchical Clustering, Spectral Clustering, etc.
Subsequence Clustering	Clustering of segments from a single long time series	Symbolic Aggregate Approximation (SAX), Sliding Window Average Binning (SWAB), Bag-of-Patterns (BOP), etc.
Time Point Clustering	Clustering of time points based on their temporal proximity and value similarity	Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Ordering Points To Identify the Clustering Structure (OPTICS), Statistical Information Grid (STING), etc.
Shape-Based Clustering	Clustering of time series based on their shape similarity, regardless of time points	Dynamic Time Warping (DTW), Longest Common Subsequence (LCSS), Edit Distance with Real Penalty (EDR), Elastic ReCovery Phase (ERP), etc.
Feature-Based Clustering	Clustering of time series based on their extracted features	Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), Piecewise Aggregate Approximation (PAA), Singular Value Decomposition (SVD), etc.
Model-Based Clustering	Clustering of time series based on their fitted models	Autoregressive (AR), Autoregressive Moving Average (ARMA), Hidden Markov Models (HMM), etc.

Table 2.1: Summary of Time Series Clustering Methods

Method	Description
Euclidean Distance	Measures the straight-line distance between two points
Dynamic Time Warping (DTW)	Aligns two time series to find the optimal match
Longest Common Subsequence (LCSS)	Finds the longest subsequence common to two sequences
Cosine Similarity	Measures the cosine of the angle between two vectors
Pearson Correlation Coefficient	Measures the linear correlation between two variables
Edit Distance	Measures the minimum number of operations to transform one string into another
SAX (Symbolic Aggregate Approximation)	Converts time series into a sequence of symbols for comparison

Table 2.2: Different Similarity Measures for Time Series Data

2.3 Multimorbidity

Multimorbidity refers to the presence of two or more chronic conditions in the same individual. Due to an increasingly aging population and increased life expectancy, the rates of multimorbidity are expected to rise throughout the 21st century [45]. In high-income countries, around 20% of people have multiple chronic conditions before the age of 40, and this percentage rises to nearly 70% for those above the age of 60 [46]. In the UK, it is projected that by 2035 around 17% of the population will have complex multimorbidity, defined as four or more co-occurring chronic conditions. This increasing prevalence will impact both individuals and healthcare services, as multimorbidity is predictive of increased mortality [47], greater functional decline [48], and a substantial economic burden on healthcare systems [49].

Several risk factors have been identified for developing multimorbidity, including increasing age [50], poor socioeconomic status [51], high body mass index, and smoking [52]. Since multimorbidity develops over a long period of time, cross-sectional studies have limited utility in analysing it. Conditions often influence the occurrence of other diseases, and these complex interactions are rarely captured in cross-sectional analysis.

Various techniques have been implemented to identify clusters of multimorbidity conditions and patient trajectories. This section presents studies on multimorbidity, its risk factors, methodologies, and the most common results. The focus is also on studies that investigate the long-term progression of multimorbidity.

Clustering, as a technique to identify trends in patients with multimorbidity, helps find groups of diseases or patients with similar conditions or trajectories. Cluster analysis can reveal important associations between conditions, such as in [2], where clusters were identified based on Yule's Q measure of association. These results provided insights into which conditions co-occurred at higher rates than expected, suggesting potential interactions between diseases.

Multiple studies have applied different clustering algorithms to multimorbidity datasets. For instance, [53] and [54] used multiple correspondence analysis (MCA) followed by k-means clustering to identify patient groups based on co-occurring conditions. Another approach was taken by [55], who used fuzzy c-means clustering on the Swedish SNAC-K cohort to allow individuals to belong to multiple clusters, reflecting the complexity of clinical presentations. Unlike k-means clustering, which assigns each data point to a single cluster, fuzzy c-means

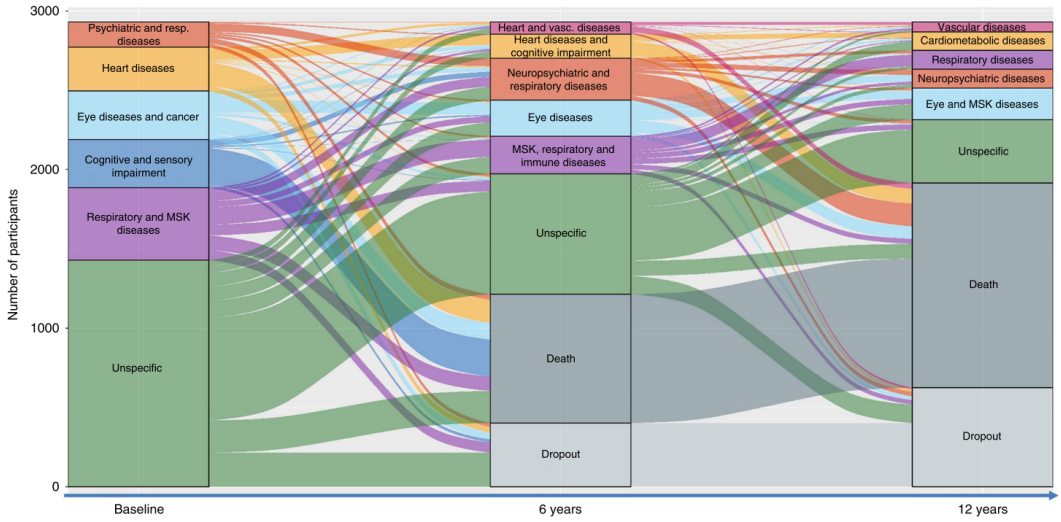

Figure 2.6: Image from [3] showing the progression of participants between clusters for 3 time points.

is more suitable in clinical contexts where patients may exhibit characteristics that span multiple disease clusters [56].

Other studies have applied fuzzy c-means clustering to investigate the SNAC-K dataset specifically for analysing multimorbidity progression over time. For example, [3] examined the progression of multimorbidity over 12 years, with clusters identified at baseline, 6 years, and 12 years. Subjects were categorised based on how they transitioned between these clusters, providing insights into disease progression dynamics. The clusters and their transitions are presented in Figure 2.6.

In contrast to patient clustering, some studies have focused on clustering conditions. [57] used agglomerative hierarchical clustering to find clusters of diseases among elderly patients in the REPOSI study, while in [58] researchers employed hierarchical clustering on Austrian EHR data to investigate the transitions between disease clusters.

To explore longitudinal multimorbidity clusters, recent studies such as [6] have reviewed multimorbidity trajectories. In [59], finite mixture modelling was used to identify six distinct clusters of comorbidity trajectories among approximately 41,000 patients. Similarly, other longitudinal studies have used techniques like growth mixture modelling [60] and group-based trajectory modelling [61] to capture changes in multimorbidity over time.

Latent Class Analysis (LCA) is a statistical method used to uncover unobserved subgroups (latent classes) within a population based on patterns of categorical or binary variables. In the context of multimorbidity, these variables often represent the presence or absence of chronic conditions in individual patients. LCA assumes that the co-occurrence of conditions can be explained by membership in a finite number of latent classes, with the conditions being statistically independent within each class. A strength of LCA in multimorbidity research is its ability to work directly with categorical diagnosis data without the need for distance metrics, making it well-suited for EHR-based studies. LCA was used in [62] to identify multimorbidity trajectories among U.S. veterans, providing valuable insights into the co-

occurrence of conditions over time. Latent class growth analysis (LCGA) [63] has been applied to UK longitudinal patient data to identify subpopulations with distinct multimorbidity trajectories, revealing differences in both baseline disease burden and the subsequent rate and pattern of condition accumulation over time [64]. LCGA is a statistical method that extends latent class analysis to longitudinal settings, grouping individuals into latent classes based on similarities in their trajectory shapes over repeated measurements.

2.4 Application of Machine Learning to Multimorbidity and Healthcare

Structured data, such as that stored in electronic health records (EHR), is essential for its primary purpose — accurately recording, storing, and retrieving patient information in a consistent format for clinical and administrative use. However, when such data is used for secondary purposes, such as advanced longitudinal research into multimorbidity progression, certain limitations can arise. For example, structured formats may encode diagnoses as isolated events without explicitly capturing temporal relationships, disease trajectories, or the context of comorbidity development. While structured data excels at supporting point-of-care decisions and standardised reporting, its design can make it challenging to directly extract the sequential patterns and complex dependencies needed for trajectory modelling without substantial preprocessing and transformation. EHR data can be categorised in the following way:

1. **Structured Data** is comprised of database tables where information such as demographic data (e.g., birth date, race, ethnicity, education), admissions and discharges to hospitals, diagnosis codes, medications, laboratory results, and allergies are stored.

    (a) **Downsides of structured data** While this type of structured tables are ideal for machine learning and data analysis, they are impractical. This type of data storage required anticipating all possible types of data and also results in sparse tables where most data categories are not present for patients.

2. **Unstructured data** Unstructured data refers to clinical notes. These are mostly in the form of free text. There is a wide range of uses for this type of free text data (radiology report, surgical note, discharge note). Unstructured textual data represents around 80% of data in EHRs [65]. This type of patient data is most flexible as it is simple to represent a wide range of information, such as medical history, family history, lifestyle data, risk factors, and treatment results.

    (a) **Downsides of unstructured data** Although clinical notes provide flexibility in representing patient data, they also have their own unique challenges. The structure of these notes is up to the doctors or nurses who are writing them. This results in clinical notes that lack structure or a systematic framework, and in turn makes them difficult to process and use for research purposes. In addition to potential misspellings, these notes can also have short phrases, abbreviations, and local dialects, thus making them even harder to process [66]. More details about NLP applications to clinical decision support are presented in [67].

EHR data poses significant research problems as it is often incomplete, irregular in time, heterogeneous, and fragmented over multiple institutions [1]. These problems arise from the fact that EHRs are used both for clinical records as well as billing purposes.

2.4.1 Mining Patterns in EHR

This section presents different classes of methods applied to EHR data analysis. For each class of methods, studies implementing these methodologies are explained.

2.4.1.1 Pairwise Correlation

Pairwise correlation studies investigate the "connection" between diseases. If 2 conditions co-occur at higher frequencies than their individual rates of occurrence, then they are considered "connected". In [68], these correlations were used to construct graphs where nodes represented diseases and their edges - "connectivity" between diseases.

This approach was extended in [69], where the authors included time difference between the incidence of diseases and the connectivity between diseases in the edges. [70] also used co-occurrence frequency as well as dynamic time-warping to find the similarity measure to find clusters of multimorbidity. In a later study, the same authors extended their method to also include genetic information [71]. Pairwise associations and network elements were also used in [72] to design MORBINET: a publicly available visual network of type 2 diabetes mellitus comorbidities.

However, pairwise methods are limited, due to them being unable to represent conditional probabilities between multiple diseases. [73].

2.4.1.2 Factorization methods

In multimorbidity research, patient health records can often be represented as a patient–condition matrix X, where each row corresponds to a patient, each column corresponds to a clinical feature (e.g., a diagnosis, laboratory measurement, or medication), and each entry represents the presence, absence, or severity of that feature for the patient. This matrix can be binary (e.g., 1 for presence, 0 for absence) or contain weighted values (e.g., frequency counts, test results). Such a representation provides a compact yet comprehensive way to store and analyse complex health data [74].
Phenotyping in this context refers to identifying latent, clinically meaningful subgroups of patients based on co-occurring patterns in their health data. These phenotypes are not predefined but emerge from statistical analysis, often representing multimorbidity clusters such as "cardiometabolic profile" (e.g., diabetes, hypertension, obesity) or "respiratory multimorbidity" (e.g., COPD, asthma, recurrent pneumonia) [75, 76]. Matrix factorization methods aim to uncover such latent structures by decomposing the original patient–condition matrix into two lower-dimensional matrices:

$$X \approx WH,$$

where W is the *patient–factor matrix* (indicating how strongly each patient expresses each phenotype) and H is the *factor–condition matrix* (indicating which conditions are associated with each phenotype).
Matrix and tensor factorisation techniques have been widely applied to Electronic Health Record (EHR) data to investigate multimorbidity patterns. In [75], a non-negative matrix factorisation technique called Limestone was introduced to produce clusters of patients, each corresponding to shared disease patterns. Later, [76] extended this model to include medications as an additional dimension, and further enhancements incorporated procedures as well.
Granite [77] was a further improvement on the Marble framework, which allowed for the discovery of phenotypes that had less overlap and would be more informative to clinicians.

Temporal EHR data has also been addressed in factorization methods. For example, [78] introduced a matrix factorization technique where one dimension of the EHR matrix was the time-axis. The derived phenotypes however, were only used to investigate Congestive Heart Failure prediction and not multimorbidity.

Another study [79] involved tensor factorization PARAFAC2, where the authors found temporal phenotypes from medically complex children. In [80], the researchers created a tensor with time being one of the dimensions and extracted phenotypes that helped predict the onset of cardiovascular diseases. [81] has recently introduced another factorization method based on PARAFAC2 that accounts for both dynamic information within EHR as well as the static demographic information. These techniques could be applied to investigate multimorbidity phenotypes. Furthermore, [82] constructed matrices of patient information and concatenated them along the time dimension. This technique was shown to be useful in generating new multimorbidity phenotypes.

While factorization techniques offer tools to find latent factors for patients, the resulting latent factors are not easily interpretable. Additionally sequential datasets must be introduced by significantly altering the original factorization methods.

2.4.1.3 Probabilistic Models

In [83], the authors highlight the advantages of using probabilistic machine learning models in healthcare, particularly their ability to estimate full predictive distributions over possible medical outcomes rather than providing only a single point estimate. For instance, a survival model may predict that two patients have the same *expected* survival time (i.e., identical predicted means), but with significantly different *uncertainties* (variances) around those predictions. This difference reflects the model's confidence in each prediction and can substantially influence clinical decision-making. For example, a treatment plan might differ if one patient's prognosis is highly uncertain while the other's is relatively precise. In addition, probabilistic models can naturally handle common challenges in EHR data such as missing values [84] and censoring [85]. A number of probabilistic approaches have been applied to study EHR data.
For example, [86] used free-text notes, medication orders, diagnosis codes, and laboratory tests to find probabilistic disease phenotypes. However, each data type is treated as a bag of

words; therefore, all potential temporal information is not utilised.

In a related work, [87] introduces a composite mixture model, which allowed to model different types of variables using different appropriate distributions. Using this approach, the authors were able to identify distinct 20 clusters of high mortality risk sepsis patients. The authors did not use time-variable features in their modelling; instead, they opted to use representative statistics of the time series.

[88] cluster pediatric ICU time-series by first discretising time into hour-long intervals using Piecewise Aggregate Approximation (PAA). PAA first splits a time series into non-overlapping intervals, then computes the mean values for each frame. The PAA approximation is then assembled from these mean values. A Gaussian mixture model was then used to cluster time series with an informative prior distribution placed on the mean parameters to deal with sparseness in the dataset. The resulting posterior distribution represented the cluster memberships for each patient.

[89] developed an unsupervised Bayesian framework titled MixEHR to find meaningful disease topics. Their approach is related to the LDA [90] as well as the matrix factorization techniques discussed in the previous section. The contribution of their method is modelling multiple types of EHR data simultaneously. The inferred phenotypes are also used for missing data imputation, a common problem in EHR data mining [66]. However, the model does not take into account the sequence of EHR data, treating each data type as a bag of words. The optimal number of clusters was identified using cross-validation and the predictive likelihood.

Selecting the number of clusters or phenotypes is commonly done by finding the optimal value of some metric (purity, predictive likelihood [89], Bayesian information criteria (BIC) [87]) and then choosing the number of clusters that produces the optimal value of the chosen metric. The Dirichlet Process, a Bayesian non-parametric method, is a stochastic process that updates the number of clusters as more data is gathered. Applying this to EHR phenotyping would allow us to investigate how the number of clusters differs based on the current sample.

2.4.1.4 Hidden Markov Models

Modelling longitudinal electronic health record (EHR) data is essential for understanding disease progression, supporting prediction, and enabling clustering of patient trajectories. While recent deep learning models can capture complex temporal dependencies, they typically require large, diverse datasets to avoid overfitting [91–93]. In many healthcare settings, such data are scarce, noisy, and irregularly sampled, making more data-efficient probabilistic approaches attractive.

State-space models (SSMs) are a family of probabilistic models that describe sequential data by introducing latent (unobservable) states, which evolve over time and emit observable measurements according to a probabilistic distribution. When these latent states are discrete, the model is referred to as a *Hidden Markov Model* (HMM). An HMM extends the concept of a *Markov chain*—which models transitions between observed states—by allowing the underlying state sequence to be hidden, with only indirect, noisy observations available. This makes HMMs particularly well-suited for disease modelling, where the true disease state is not

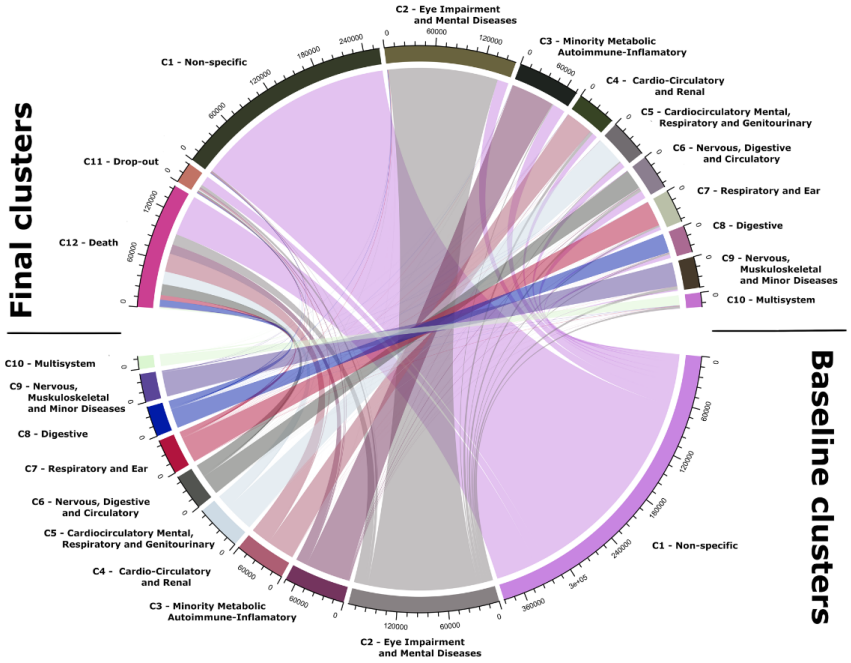

Figure 2.7: Image from [94] showing the patient transitions from baseline to final clusters. Clusters are characterised by the most prominent conditions within a cluster.

directly observable, but related clinical signals (e.g., biomarkers, diagnoses) can be measured. In [94], an HMM model was used to model the longitudinal trajectories of participants over the course of 5 years. Their process involved first reducing the dimensionality of the data. Then apply k-means to the reduced dataset to set the initial HMM parameters. The latent states of the HMM were treated as clusters of diseases, characterised by the most prominent disease in a cluster. Resulting transitions from baseline to final clusters from [94] are displayed in Figure 2.7. The authors found that close to half of the subjects at baseline were part of a non-specific cluster, where no single disease was over-represented.

Weighted Association Rule Mining (WARM) is an extension of traditional association rule mining (ARM) that incorporates item-specific weights to better reflect their relative importance in the domain of interest [**ma2014weighted**]. Standard ARM techniques, such as the Apriori algorithm, treat all items equally when computing support and confidence, meaning that frequent but clinically trivial itemsets can dominate the results. WARM addresses this limitation by assigning each item (diagnosis, medication, or symptom) a weight that represents its significance, which may be derived from clinical impact, prevalence, cost, or other domain-specific considerations. In the healthcare context, this allows rare but clinically critical conditions to contribute more strongly to rule generation than common but less consequential conditions.
In [95], WARM was applied after estimating disease transition probabilities using first-order Markov chains. Here, WARM was used to discover clusters of chronic conditions that fre-

quently co-occurred, with weights reflecting the relative clinical importance of each condition. This dual-step approach combined temporal modelling (via Markov chains) with weighted co-occurrence mining (via WARM), resulting in a richer characterisation of multimorbidity patterns than would be possible with either method alone. The authors note, however, that although their model takes into account the progression of diseases, it is limited in multiple ways. First, they only use first-order Markov chains, meaning that the probability of a future condition is only dependent on the previous time step and not the entire history. The solution for this would be to include higher order Markov chains; however, the authors chose not to do this as interpreting all combinations of 103 chronic conditions would be too complicated. Secondly, although this approach uses the sequences of events to find clusters of diseases, the time intervals between diagnoses are not taken into account.

A continuous-time hidden Markov model (CT-HMM) was introduced in [96] to model patient disease trajectories through hidden states. Using this model, the likelihood of a given journey can be calculated, which can be used to find a similarity between different patient trajectories. A mixture model was then used to cluster patients and found subtypes of progression of hemodynamic instability. However, the distance was only calculated using the likelihood of a particular trajectory. The distance between the HMM models themselves was not identified. Identifying the distance between models would give a more thorough method of comparing patient trajectories, based on their latent state sequence.

Another study used Bayesian HMMs (B-HMM) to find clusters of patients from Chinese EMR data spanning 9 years [97]. They first segmented each patient's health records into sets of epochs of treatment events. Then their approach involved finding latent treatment topics for each epoch and transitions between topics. They used a B-HMM to represent patient trajectories with Dirichlet priors placed on the transition parameters and emissions distributions. Agglomerative hierarchical clustering was used with the likelihood of patient journeys being used as a similarity measure.

HMMs were also used to cluster patient medical trajectories in [98]. In their approach, they used both categorical variables (diagnosis codes) and continuous variables (vitals and laboratory test results) for clustering. They first mapped each medical trajectory to an HMM and then used KL divergence to compute the distances between HMMs.

In [99], the longest common sub-sequence was used as a distance metric to cluster patients. A first order HMM was then trained to find the most probable trajectories using time-series of multiple laboratory test observations and other patients' characteristics. They were able to identify the most probable clinical trajectories from chronic kidney disease patients.

The authors of [100] used an autoregressive HMM to learn shared dynamics from time series measurements in several studies, [101], [102]. The aim was to find recurring time-series segments across the patients. Their model was able to improve mortality prediction and sepsis detection.

An example time-series clustering using HMMs not related to EHR data is presented in [103].

The authors first used Dynamical Time Warping to calculate distances between driver time series. These distances are then clustered using hierarchical clustering to identify subsets of driver behaviour. These subsets of driver data are used as a sequence of inputs into an HMM. This HMM is then used to iteratively optimise the procedure until convergence is achieved and the optimal clustering of drivers is found.

A method of efficiently training Continuous-Time HMM was introduced in [104]. Traditional HMMs assume that measurements are observed at regular discrete intervals, which is not the case for EHR data, due to missed visits and other complications. The authors modelled disease progression using CT-HMMs.

There have been recent adaptations of the traditional HMM model to complex health data. In [105], a coupling between conditions is introduced. In the paper, the authors present a coupling between diabetes and liver disease, which is represented by a modified transition matrix. Additionally, the authors differentiate between the severity of conditions by using different states for acute and stable disease states. The authors show that including coupling, having static patient attributes influence the transitions and introducing different disease states produced better results than a regular HMM model.

One important limitation of Hidden Markov Models is their assumption that a patient's future disease trajectory is determined solely by their current state. This "memoryless" property prevents HMMs from capturing the heterogeneity in patient trajectories that often arises from complex and varying clinical histories [106]. In such models, all patients in the same latent state share identical transition probabilities to other states, regardless of their previous conditions or other influencing factors. This can be particularly problematic in multimorbidity research, where disease progression is frequently shaped by earlier chronic conditions. To overcome this issue, [106] introduces an attentive state space model that incorporates a patient's entire history into the state transition process, allowing for more nuanced modelling of individual disease pathways.

Chapter 3

Sequence Clustering Pipeline

This chapter presents the sequence clustering pipeline developed for clustering multivariate time-series data. The pipeline is comprised of two stages to identify complex patterns within the data Figure 3.1. Multivariate time-series data refers to observations of multiple variables recorded simultaneously at regular intervals over time. This pipeline specifically requires that these observations be made at equally spaced time intervals. Stage 1 of the pipeline first clusters individual timepoints, grouping together those that exhibit similar patterns across all sequences. This reduces the high-dimensional multivariate sequences into simplified univariate sequences based on cluster memberships. In Stage 2, these relabelled sequences are clustered using the DBSCAN algorithm, which relies on a distance measure called State Space Dynamic distance. This distance between sequences is computed with the aid of a Hidden Markov Model (HMM), which helps capture the temporal dependencies and transitions between different states within the sequences. While the clustering itself is performed by DBSCAN, the use of the HMM to compute the distance ensures that the temporal structure of the sequences is taken into account, allowing for more accurate clustering of sequences based on their underlying dynamics. The pipeline, along with the optimization of the various hyperparameters involved, is displayed in Figure 3.2.

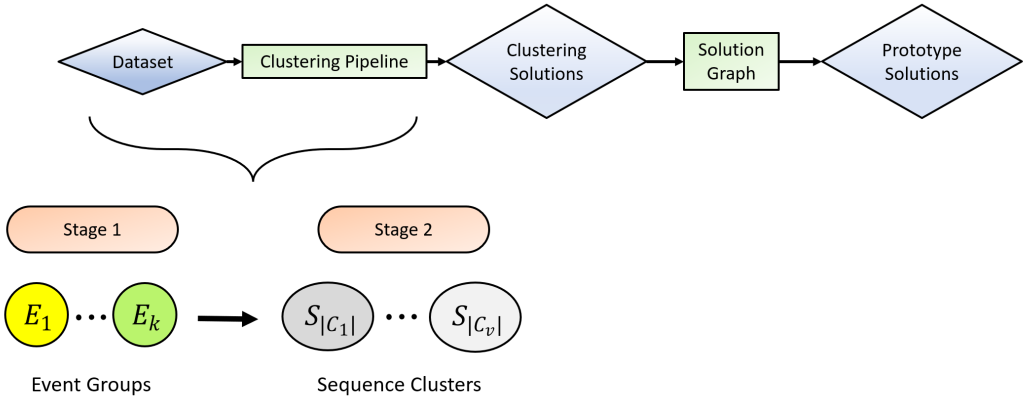

Figure 3.1: Computation diagram for this chapter, the final outcome of this chapter being event groups and the sequence clusters.

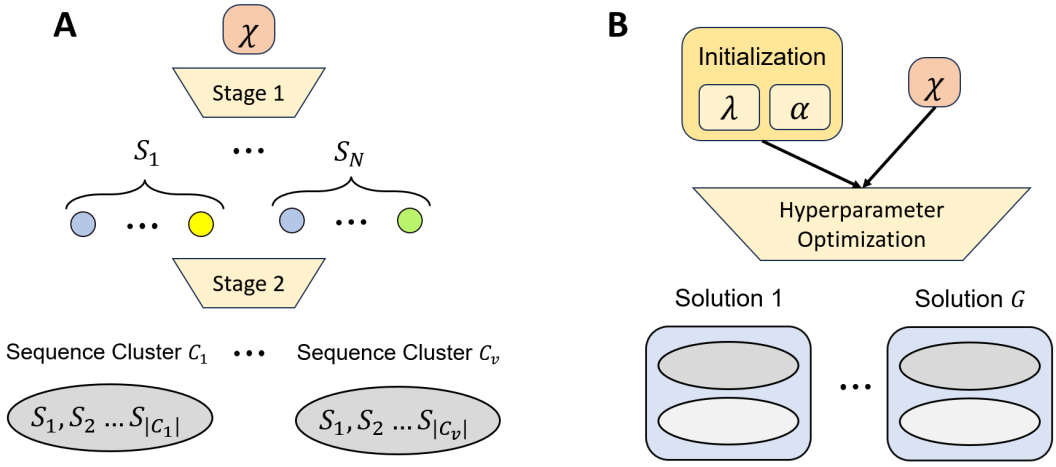

Figure 3.2: Part A) of the diagram is a visual representation of Stage 1 and 2 of the pipeline, where in Stage 1, the dataset of multivariate time series timepoints is converted to sequences of cluster memberships. Stage 2 of the pipeline then clusters the sequences using a state-space model. Part B) of the diagram is a representation of the hyperparameter optimisation operation, where λ and α are initialization parameters for the optimisation, and the output consists of a total of G solutions.

3.1 Stage 1 - Timepoint Clustering

To tackle the problem of clustering multivariate time-series we use a strategy of first simplifying the dataset by introducing the concept of **Event Groups**. Multivariate time-series are often difficult to interpret due to their high-dimensional nature [107]. The aim of Stage 1 of the pipeline is to reduce the dimensionality of the dataset, while maintaining the relevant information stored.

Each timepoint in the dataset is assigned to an event group during Stage 1 of the pipeline. An Event group represents a collection of commonly co-occurring events within the dataset. In addition to simplifying the dataset for further clustering, this method also aids in the interpretation of later sequence clusters. At the end of this first step, a multivariate time-series is represented as a sequence of event group memberships, which in turn correspond to trends of events within the original dataset.

In this section, we describe each step of Stage 1 of the Sequence Clustering pipeline. These steps involve converting a multivariate time series dataset into a sequence of Event Group memberships, later clustered in Stage 2 of the pipeline. This procedure is visualized in Figure 3.3.

3.1.1 Event Groups

Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T]$ represent the multivariate time series of T timepoints. Each timepoint $\mathbf{x}_t$ at time t is a binary column vector of length D, where D corresponds to the number of possible events in the time series. Timepoint $\mathbf{x}_t$ is defined as:

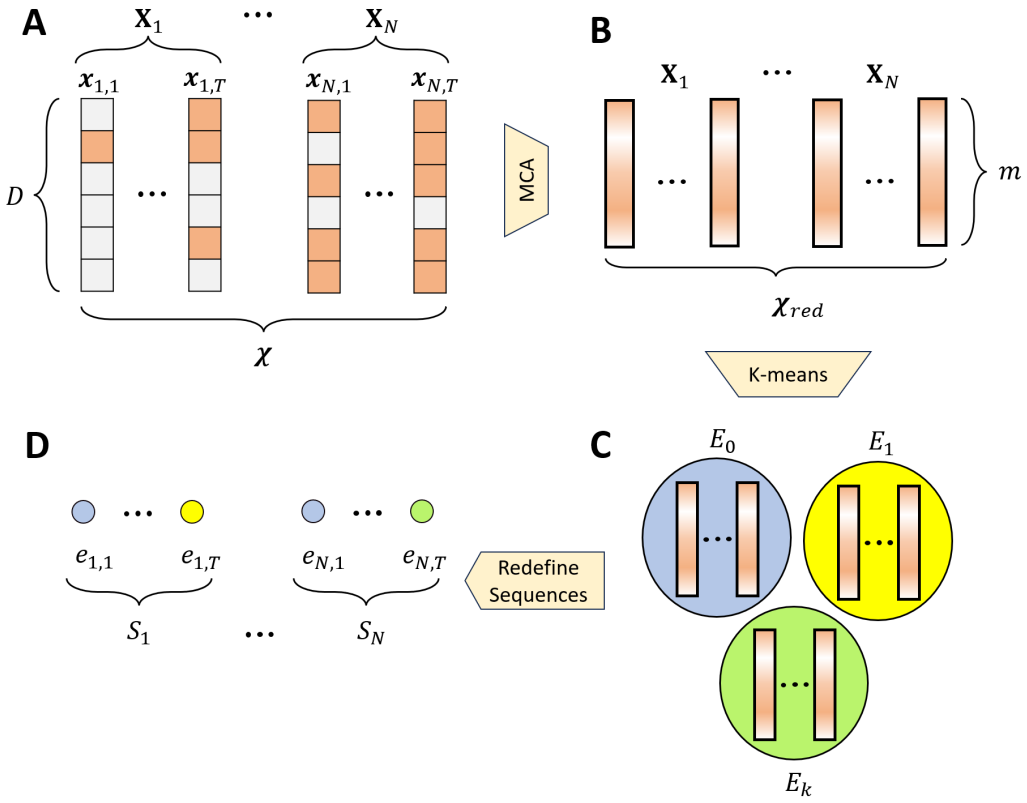

Figure 3.3: Diagram of Stage 1 of the clustering pipeline. Part A) is the original dataset of multivariate sequences. Part B) shows the process of reducing the dimensionality of the dataset using MCA. Part C) is the next step in Stage 1 of the pipeline where individual timepoints are clustered into Event Groups. Lastly, Part D) are the redefined multivariate sequences of Event Group memberships.

$$\mathbf{x}_t = (x_{1,t}, x_{2,t}, \ldots, x_{D,t}) \tag{3.1}$$

where $x_{d,t}$ represents the value of the d-th variable at time t, and $d = 1, 2, \ldots, D$. A single time series consists of a total of T observations. In the case of multiple time series, let $\mathcal{X}$ represent the dataset containing N multivariate time series,

$$\mathcal{X} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N], \tag{3.2}$$

where $\mathcal{X} \in \{0, 1\}^{D \times NT}$ and where each column corresponds to a different timepoint from a different time series, and each row corresponds to a different event. Individual entries of $\mathcal{X}$ are binary values in a matrix form,

$$\mathcal{X} = \begin{bmatrix} x_{1,1}^1 & \cdots & x_{1,t}^n & \cdots & x_{1,T}^N \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{d,1}^1 & \cdots & x_{d,t}^n & \cdots & x_{d,T}^N \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{D,1}^1 & \cdots & x_{D,t}^n & \cdots & x_{D,T}^N \end{bmatrix}, \tag{3.3}$$

where $n = 1, 2, \ldots, N$ and N is the total number of time series in the dataset. $d = 1, 2, \ldots, D$, where is D the total number of unique events and $t = 1, 2, \ldots, T$ are the total timepoints in the dataset. An example of a multivariate time series matrix is presented in Figure 3.4. We use this matrix format of columns of individual observations to organise the sequences for the following clustering stages.

	Sequence 1					Sequence 2				
	t1	t2	t3	t4	t5	t1	t2	t3	t4	t5
Event 1	1	1	1	1	0	0	0	0	0	0
Event 2	0	0	0	1	0	1	0	0	0	1
Event 3	1	1	1	0	0	0	0	1	0	1
Event 4	1	1	1	1	1	0	0	1	1	1

Figure 3.4: An example showing how multivariate time series data is ordered in a single matrix. The matrix consists of 2 sequences along with 4 possible events and 5 timepoints.

After this initial reordering of the dataset, we proceed to clustering each column to identify similar groups of timepoints. These clusters of timepoints are defined as **Event Groups**.

First, Multiple Correspondence Analysis (MCA) [108] is used to transform the high-dimensional multivariate time series data into a condensed, lower-dimensional representation, effectively capturing the timepoints of sequences. The procedure begins by transforming the original categorical data into a complete disjunctive (indicator) matrix, where each possible category of each variable is represented as a binary column. MCA then compares the observed indicator matrix to what would be expected under the *independence model*, where all variables are assumed statistically independent and joint category frequencies equal the product of their marginal frequencies. This centering step ensures that MCA captures deviations from independence (genuine associations between categories, rather than random co-occurrence). MCA then performs a singular value decomposition (SVD) on the centered and normalized matrix (or equivalently on the Burt matrix) to identify the principal dimensions that best explain the variance, or inertia, in the dataset.

MCA is particularly well-suited to our pipeline because it simultaneously handles multiple categorical variables and maps these into the same low-dimensional space. This preserves interpretability: timepoints that share many events will be mapped close together. By reducing the dimensionality, MCA removes noise and highlights latent structures that aid in subsequent clustering.

The dimensionality reduction changes the data from a binary matrix to a matrix of scalar values $\mathcal{X}_{red} \in \mathbb{R}^{m \times NT}$, where $m < D$. Subsequently, the reduced-dimensional data undergoes K-means clustering, a popular unsupervised learning algorithm, to delineate distinct clusters of timepoints. For instance, if two timepoints share a significant portion of the same events, they are considered similar and are likely to be part of the same timepoint cluster.

After applying MCA, all timepoints in $\mathcal{X}_{red}$ are clustered using K-means to find event groups. We chose K-means for its simplicity and efficiency in partitioning the data into distinct clusters based on their similarity. Given the dataset of sequences $\mathcal{X}_{red}$, K-means clustering aims to partition the dataset of size N into $k \leq N$ clusters $\mathbf{E} = \{E_0, E_1, \ldots, E_k\}$. The clusters of timepoints, denoted E_k, are composed of timepoint vectors indexed by their original time positions. If cluster E_k contains the timepoints with original indices $t_1, t_2, \ldots, t_{|E_k|}$, then:

$$E_k = \{\mathbf{x}^{(t_1)}, \mathbf{x}^{(t_2)}, \ldots, \mathbf{x}^{(t_{|E_k|})}\},$$

where $\mathbf{x}^{(t)}$ is the timepoint vector at original time index t, E_k is the k-th timepoint cluster, and $|E_k|$ is the total number of timepoints in that cluster. Groups of timepoints represent sets of similar events occurring at different positions in the sequence. After clustering, each timepoint vector $\mathbf{x}^{(t)}$ is redefined in terms of the membership of the cluster E_k to which it was assigned.

Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T]$ be the original sequence of timepoint vectors. After timepoint clustering, each timepoint vector $\mathbf{x}_i$ is assigned to a cluster E_k. Let e_t denote the cluster membership of $\mathbf{x}^{(t)}$, where $e_t = k$ if $\mathbf{x}^{(t)} \in E_k$. The sequence of cluster memberships is:

$$e_t = k \quad \text{if} \quad \mathbf{x}^{(t)} \in E_k \quad \text{for} \quad t = 1, 2, \ldots, T,$$

and the final output is the sequence of cluster memberships:

$$S = (e_1, e_2, \ldots, e_T).$$

An example of a multivariate time series matrix after clustering timepoints into event groups is shown in Figure 3.5.

	Sequence 1					Sequence 2				
	t1	t2	t3	t4	t5	t1	t2	t3	t4	t5
Event 1	1	1	1	1	0	0	0	0	0	0
Event 2	0	0	0	1	0	1	0	0	0	1
Event 3	1	1	1	0	0	0	0	1	0	1
Event 4	1	1	1	1	1	0	0	1	1	1
EG	0	0	0	0	1	2	2	1	1	0

Figure 3.5: An example showing how multivariate time series data is ordered in a single matrix along with the Event Group memberships (EG).

The multivariate time series were transformed from a difficult-to-interpret, high-dimensional dataset into a univariate sequence of Event Group memberships. In high-dimensional datasets, many timepoints often exhibit very similar or even identical event patterns, leading to substantial redundancy [107]. Clustering these redundant timepoints into aggregated Event Groups helps suppress noise, reduce dimensionality, and emphasize more meaningful temporal transitions.

3.1.2 Stage 1 Output

The aim of the Event groups is to first reduce the complexity of multivariate time-series data and enable easier interpretation. To understand the composition and trends captured in the Event groups, we introduce an Event Group Score to measure the importance of each unique event in the dataset. This score measures which events occur most frequently within an event group and the events they are paired with, providing an intuitive description of the trends within the partition of timepoints.

3.1.2.1 Event Group Score

To better understand the specific events present in each timepoint cluster, we developed an Event Group Score. This measure provides a quantitative measure of which events are most frequent within a cluster of timepoints. After the K-means clustering of $\mathbf{M}_{red}$ the resulting clusters $\mathbf{E}_k$ are groups of timepoints $\mathbf{x}_i$. Since the total occurrence of events in a single timepoint cluster depends on the distribution of event frequency in the entire database $\mathcal{M}$. Every value of x_t contains up to D total events; however, some of these event types may be overrepresented in $\mathcal{M}$, so counting how many occur in the time-point clusters will introduce bias towards the most prominent ones. The clusters are instead interpreted by introducing an Event Group Score (EGS) for event type D in timepoint cluster k:

$$EGS_d^k = \frac{B_D^k}{B_{tot}^k}, \tag{3.4}$$

where B_D^k is defined as $\frac{b_D^k}{b_{tot}^k}$ (b_D^k: number of occurrences of event D in cluster k,b_{tot}^k: total number of timepoints in timepoint cluster k), and B_{tot}^k is defined as $\frac{b_D^{tot}}{b_{tot}}$ (b_D^{tot} number of occurrences of event D in entire dataset $\mathcal{M}$ and b_{tot} is the total number of timepoints in the dataset). The purpose of this measure is to balance the number of occurrences of events in the timepoint clusters with the occurrences in the entire dataset. A larger EGS for event type D in timepoint cluster k that event D occurs at a higher rate in the timepoint cluster than the entire dataset $\mathcal{M}$. Whereas if an event occurs many times in a timepoint cluster, but also many times in the entire dataset, it is not unique to that particular timepoint cluster. This would then be reflected by a lower EGS. The use of EGS helps understand what event types are more prominent in each timepoint cluster. For comparability across clusters, the EGS for each timepoint cluster was *L2-normalised* before further use, treating $\mathbf{EGS}^k = (EGS_1^k, \ldots, EGS_D^k)$ as a vector and computing $\widehat{\mathbf{EGS}}^k = \mathbf{EGS}^k / \|\mathbf{EGS}^k\|_2$; this preserves relative weights while scaling the overall magnitude to 1.

After performing the timepoint clustering described in the previous section, we rank the importance of individual events based on the event group score. The event group score

measures the relative importance of each condition in a timepoint cluster based on the number of occurrences in the entire dataset and the specific timepoint cluster. This score simplifies the interpretation of the timepoint clusters and facilitates understanding which conditions tend to co occur in the timepoints which are clustered together. An example of a single timepoint cluster is shown in Figure 3.6. Part A of the figure shows a table of the toy dataset with 2 sequences and their respective observations, along with the timepoint clusters.

A

	Sequence 1					Sequence 2				
	t1	t2	t3	t4	t5	t1	t2	t3	t4	t5
Event 1	1	1	1	1	0	0	0	0	0	0
Event 2	0	0	0	1	0	1	0	0	0	1
Event 3	1	1	1	0	0	0	0	1	0	1
Event 4	1	1	1	1	1	0	0	1	1	1
EG	2	0	2	2	0	0	2	1	2	2

B

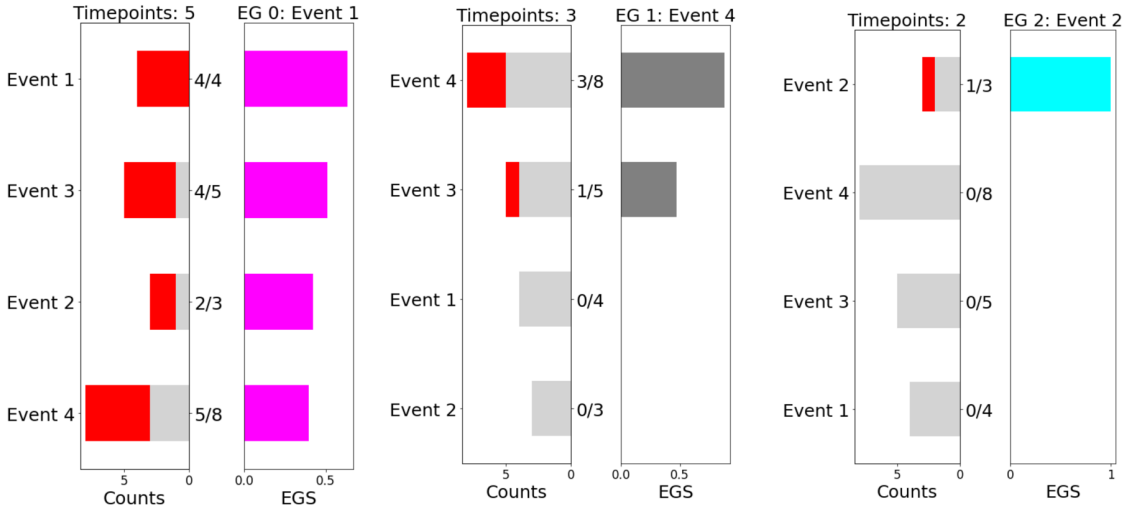

Figure 3.6: Example of the data configuration and the corresponding. Part A) Shows a table of timepoints in each row, while the columns show which event occurred at each timepoint. After the timepoint clustering stage of the pipeline, each timepoint is assigned to a timepoint cluster. Part B) shows the bar chart of timepoint clusters and their Event Group Score (EGS). On the left side, the red section of the bar shows how many counts of the specific condition are in that Disease Group, while the grey and red parts show how many observations of the specific condition were present in the entire dataset. The right-hand side shows the Event Group Score (EGS) of the event in a timepoint cluster.

An example of using the event group score to understand the Event Groups is showcased in Part B of Figure 3.6. The bar charts show the number of observations of a specific event in a timepoint cluster in red, while the entire length of the left bar chart shows how many observations of a specific event there were across the entire dataset. The right part of the

bar chart shows the EGS of a specific event in a timepoint cluster. In Figure 3.6, a high EGS score reflects the higher frequency of different events in different timepoint clusters, while taking into account how frequent these conditions were across all timepoints. For clarity in later chapters, each Event Group (EG) is referred to by the label of the event with the highest EGS.

3.2 Stage 2 - Sequence Clustering

The next stage pipeline involves clustering the sequences of Event group memberships. To accomplish this, first we chose a distance measure, the State-space dynamics distance, based on a Hidden Markov Model. This measure was chosen as it is suited for short sequences and additionally provides a probabilistic model of the sequence data, providing further insight into the trends of the dataset. The distances calculated using this measure are used as the input for the `DBSCAN` clustering algorithm, which is used to cluster the sequences. The final output of this pipeline are a combination of the Event Groups from Stage 1 alongside the sequence clusters from Stage 2, an example of the output of the pipeline is provided at the end of this section.

3.2.1 Choice of Distance Measure

The first step in clustering is choosing a distance measure. Time series are particularly difficult to cluster as the usual distance measures, such as euclidean distance, are not applicable. Distance measures between time series are discussed in further detail in the literature review.

In this subsection, the details of the distance measure used in Stage 2 of the clustering pipeline are described. Specifically, we used the State Space Dynamics distance, a distance measure for time-series, based on a probabilistic model proposed in [109]. Calculating the distance between sequences using this method involves first training a single Hidden Markov Model using all sequences in a dataset, which represents a common state-space for the dataset. The distance between sequences is then found by calculating the distances between induced transition matrices for each sequence. This distance measure addresses common problems in probabilistic modelling, such as overfitting and computational scalability. In the context of HMMs, overfitting can happen if too many states are used or if the model is trained on a limited amount of data. Previously proposed methods of sequence clustering using HMMs [110] train a single HMM on each sequence in the dataset to find a distance matrix between each element. As discussed in [109], this leads to several disadvantages, including overfitting and computational complexity, as the number of likelihoods needed to be computed is N^2. The State Space Dynamics distance addresses these issues by constraining the complexity of the HMM. The State Space Dynamics distance method uses an HMM to create a probabilistic representation of the dataset.

Let Θ be an HMM with total states Q trained using all the sequences in the dataset S. After Θ is fitted using the Viterbi algorithm [111], each sequence in S is fed through Θ using the forward-backward algorithm [37]. Each sequence S_n in S is linked to the common state space through the transition matrix that it induced when fed into the model Θ using the forward-backward algorithm. This induced transition matrix is denoted as:

$$\tilde{A}_n = \{\tilde{a}_{ij}^n\}_{i,j=1}^{Q}, \tag{3.5}$$

where $\tilde{a}_{ij}^n = p(q_{t+1}^n = s_j \mid q_t^n = s_i, \mathbf{S}_n, \Theta)$ and Q is the total states in the HMM.

Using the transition matrix $A = \{a_{ij}\}$ the sequence-specific transition probabilities are obtained using:

$$\tilde{a}_{ij}^n = \sum_{t=1}^{T-1} \alpha_i^n(t)\, a_{ij}\, p(x_{t+1} \mid q_{t+1} = s_j)\, \beta_j^n(t+1), \tag{3.6}$$

where $\alpha_i^n(t)$ and $\beta_j^n(t+1)$ are the forward and backward variables for $\mathbf{S}_n$.

After obtaining $\tilde{A}_n$ we can view the matrix as a collection Q discrete probability functions:

$$\tilde{\boldsymbol{A}}^{\boldsymbol{n}} = [\boldsymbol{a}_{n,1}, \ldots \boldsymbol{a}_{n,i}, \ldots \boldsymbol{a}_{n,Q}]^T,$$

where $\boldsymbol{a}_{n,i}$ is a discrete probability function in each row. Each row $\boldsymbol{a}_{n,i}$ corresponds to the induced transition probabilities for sequence n from state 1 to state Q.

The next step in the process is using each row of these transition matrices to find the distances between all sequences.

The distance between sequences S_i and S_j is found by calculating the distance between the induced transition matrices A_i and A_j. For this we use the Bhattacharyya affinity [112] defined as:

$$D_B(p_1, p_2) = \sum_x \sqrt{p_1(x), p_2(x)}, \tag{3.7}$$

where p_1, p_2 are arbitrary discrete probability distributions. The distance between the transition matrices A_i and A_j is found by calculating the mean affinity for each row a_n, where the rows correspond to the transition probabilities between each state and every other state. The distance between the sequences S_i and S_j can then be defined as:

$$d_{ij}^{BHAT} = -\log \frac{1}{Q} \sum_{l=1}^{Q} D_B(p_{il}, p_{jl}), \tag{3.8}$$

where Q is the total number of states in the HMM Θ.

3.2.2 Clustering Sequences Using `DBSCAN`

Once the distances between all sequences in S are calculated, they are used as inputs in the `DBSCAN` clustering algorithm. The principle of `DBSCAN` (Density-Based Spatial Clustering of Applications with Noise) is to find regions of high density that are separated from regions of low density. The `DBSCAN` clustering algorithm is used with 2 input parameters: ϵ (maximum distance between two samples for one sample to be considered being in the neighbourhood of the other sample) and *min_points* (number of samples in a neighbourhood for a point to be treated as a core point). The advantage of using `DBSCAN` is that the number of clusters does not need to be predefined as it does in K-means. An additional advantage of the DSBSCAN algorithm is that a precomputed distance matrix can be used to find clusters. Although `DBSCAN` is a powerful algorithm, there are several disadvantages to this method. Determining the parameter ϵ and *min_points* is challenging and, if chosen poorly, can lead to poor results.

Another disadvantage is the computational complexity. The time complexity for `DBSCAN` is $O(n \log n)$, which makes calculating clusters for large datasets slow. Methods such as mini-batch k-means, which achieves near -linear passes over the data by using small stochastic batches [113], or BIRCH [114], require Euclidean embeddings and typically a user-specified k for the final partitioning [20]. They also do not detect noise or arbitrarily shaped clusters by design (in contrast to density-based methods) [115].

The final output of the `DBSCAN` clustering is sequence clusters, defined as follows:

$$\mathbf{C} = \{C_1, C_2, \ldots, C_v\}, \tag{3.9}$$

where the sequence cluster is defined as C_v, v denoting the total number of sequence clusters. Each cluster C_v contains a set of sequences, indexed locally within that cluster, as follows:

$$C_v = \{S_{v,1}, S_{v,2}, \ldots, S_{v,|C_v|}\}, \tag{3.10}$$

where $S_{v,j}$ denotes the j-th sequence in cluster C_v, and $|C_v|$ is the number of sequences in that cluster. This double-index notation avoids ambiguity by explicitly linking each sequence to its corresponding cluster. A diagram of Stage 2 of the pipeline is detailed in Figure 3.7.

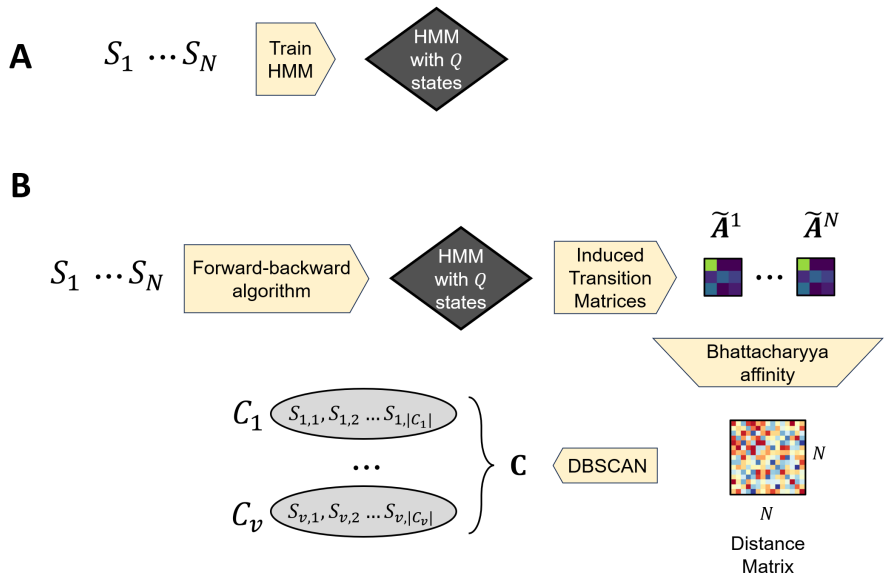

Figure 3.7: Diagram of Stage 2 of the clustering pipeline. In section A) sequences are used to train a single HMM model. In section B) sequences are passed through the HMM model, the distance between transition matrices is calculated, and `DBSCAN` is used to cluster the distance matrix to find clusters of sequences.

3.2.3 Stage 2 Output

The final output of the pipeline is the Event groups and the sequence clusters. This section introduces measures and visualisation tools used to understand and interpret the output of

both stages of the pipeline.

After completing the initial timepoint clustering process, the dataset is redefined in terms of the timepoint cluster memberships. The redefined sequences are used in Stage 2 of the pipeline to cluster the time series data. This redefining step improves the pipeline in several ways. First, it enables a more interpretable and less complex representation of the original data. Instead of reading the dataset as a multivariate time series, which makes spotting trends and tendencies difficult to non-experts, after the redefining, we are left with a univariate sequence, which is easier to read and understand. Secondly, the clustering procedure is easier to perform with a univariate time series dataset.

To visualise univariate time series of Event Group clustering labels, we use Sankey Diagrams [116]. Sankey diagrams are a useful visualization tool used to represent flows and their transformations within a system. Sankey diagrams excel in showing how data points move across timepoint. This helps in understanding the trends of cluster memberships across several time series.

An example of this using a toy dataset is presented in section A of Figure 3.8. The table shows an example of sequences with 5 timepoints and 4 possible events, either 1 or 0, corresponding to the event either occurring or not, respectively. The EG (event group) row of the table shows the results of the 1st stage of the clustering pipeline. These integers denote which timepoint cluster or event group the specific observations were clustered in using K-means. In this example, there are 3 separate timepoint clusters. The sequence data is then reorganised based on the sequence of EG memberships. This reorganisation is shown in the EG row of the table of part A in Figure 3.8. We visualise the redefined sequences and clusters of sequences using Sankey diagrams in section B.

In this example, they show the progression of the sequence timepoint cluster memberships across the 5 observations. The Sankey diagram uses distinct colours to distinguish between the different timepoint clusters. The 2 sequences can also be visualised using a Sankey diagram, shown in part B of Figure 3.8. There the individual sequences are presented using Sankey diagrams on top, along with the combined dataset of 2 sequences in a single Sankey diagram. This visualisation allows for easier interpretation of a large number of sequences, which will be shown in later sections. The height of the arrows in the sankey diagram corresponds to the total number of occurrences of a specific Event Group.

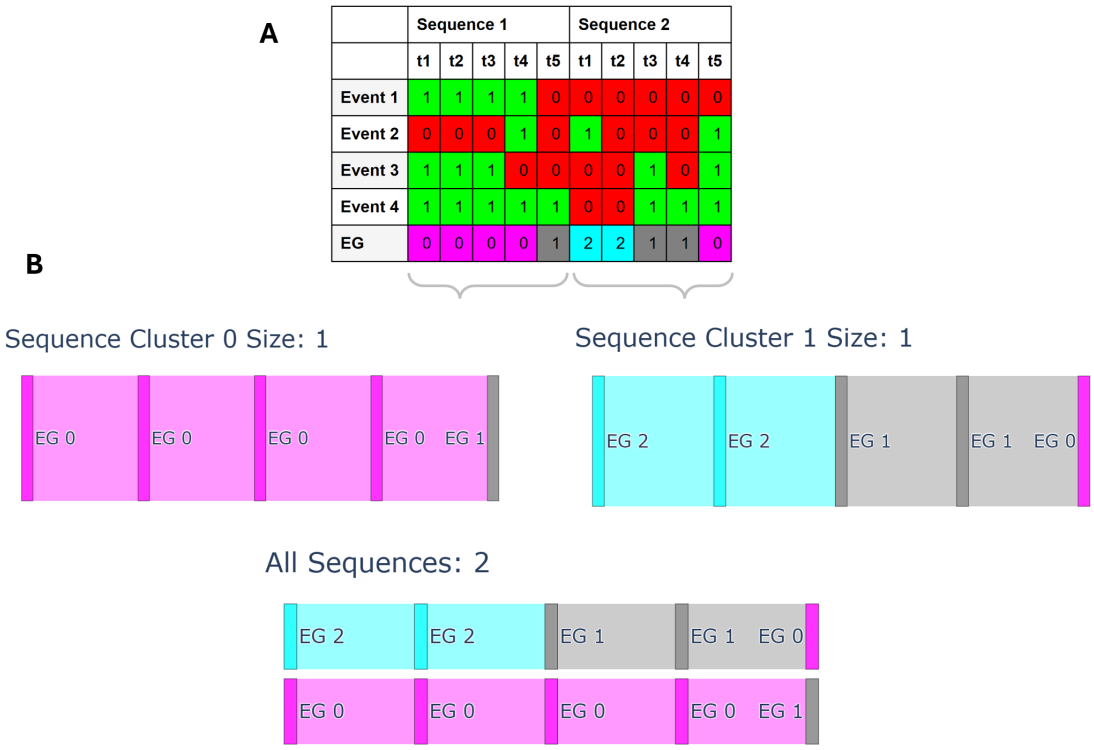

Figure 3.8: The figure shows how the EGs of the timepoints are turned into Sankey diagrams. Part A) shows a matrix of the time series dataset of Sequences 1 and 2 and their corresponding Event Group assignments (EG). Part B) shows the Sankey diagrams of the individual sequences and both sequences together.

3.3 Pipeline Optimization

Hyperparameter optimisation is a crucial aspect of refining machine learning models for optimal performance. Hyperparameters, which are set prior to training, play a pivotal role in influencing a model's learning process. Several techniques of identifying optimal hyperparameters exist, such as grid search, random search, and Bayesian optimization, which are commonly employed for this purpose [117, 118].

In clustering algorithms, the adjustment of parameters like the number of clusters or distance metrics can significantly impact the final clustering results. Efficient hyperparameter optimization ensures that models are well-suited to the specific characteristics of the data, leading to improved performance and enhanced generalization on unseen data instances.

The 2-stage clustering model contains 5 parameters: MCA dimensions (m), number of K-means clusters, number of states in the HMM, and `DBSCAN` parameters (ϵ and minimum samples). To find the best set of parameters, we used Bayesian hyperparameter optimisation, with the Tree of Parzen Estimators (TPE) hyperparameter optimisation algorithm [119] from the Hyperopt package [120]. In evaluating the quality of our clustering results, we employ the Silhouette index [26].

The silhouette index is a metric used to evaluate the quality of clustering in a dataset. It quantifies how well each data point fits into its assigned cluster relative to other clusters. The silhouette index ranges from -1 to 1, where a higher value indicates that the data point is well-clustered, with a tight cluster and good separation from neighbouring clusters. Conversely, a negative value suggests that the data point may have been assigned to the wrong cluster, while values close to zero indicate overlapping clusters. The value optimised was the sum of the silhouette indices from the timepoint clusters and final sequence clusters.

We define this combination of 2 silhouette indices as w and introduce a mixing coefficient α which prioritises either timepoint clusters or sequence clusters. This is defined in the equation below:

$$w = \alpha c_{sil} + (1 - \alpha) s_{sil}, \tag{3.11}$$

where the c_{sil} and s_{sil} are the silhouette indices for the timepoint clusters and sequence clusters, respectively. The hyperparameters were selected by minimising the negative of w, i.e., setting the loss to $L = -\mathrm{w}$, which is equivalent to maximising the combined Silhouette scores; thus, the lowest loss corresponds to the highest Silhouette combination (best cluster separation/compactness).

During our experiments, hyperparameter optimisation was run several times with different initialisation parameters λ: different numbers of rare events removed and different mixing coefficient α values. We use the combination of 2 silhouette index values from the 2 stages because we are looking for solutions which produce high-quality results across both stages of the pipeline. Optimising each stage individually was tested but resulted in subpar outputs (details in the Appendix). Tuning a single DBSCAN parameter while holding the other fixed (e.g., choose ε with `min_samples`=5, then tune `min_samples`). Because ε and `min_samples` are coupled, step-wise tuning can become trapped on ridges of the objective surface and produce weaker solutions. The solutions ofthen fail basic internal checks on the precomputed distance matrix: poor separation/compactness (low Silhouette), degeneracy (one giant cluster or extreme fragmentation), excessive noise fraction, or instability under small parameter perturbations. This simultaneous way of optimising both silhouette indices ensures that both the timepoint clusters as well as the sequence clusters are of high quality. Although this optimisation provided high-quality results in terms of the combined silhouette indices, there was a large amount of overlapping solutions. The sets of parameters which resulted in the best loss values in some cases provided largely similar solutions.

3.4 Pipeline validation

Validation is a critical step in the development and assessment of any clustering algorithm, as it ensures that the algorithm performs reliably and produces meaningful results. In this section, we focus on the validation of our sequence clustering pipeline. The validation process is designed to test the robustness, accuracy, and effectiveness of our clustering approach in identifying the underlying structure of the data.

To rigorously evaluate the performance of Stage 2 of the pipeline, we apply it to a series of artificially generated datasets. These datasets are constructed using predefined HMMs, allowing us to control the complexity and characteristics of the sequences. By using synthetic

data, we can create scenarios where the true underlying models are known, providing a clear benchmark against which to measure the success of our clustering procedure.

The primary objective of this validation process is to determine whether our sequence clustering pipeline can accurately recover the number of HMMs used to generate the artificial sequences. This involves not only identifying the correct number of clusters but also ensuring that the sequences within each cluster are associated with the same HMM model. Success in this validation task would demonstrate that our methodology is capable of capturing the essential temporal dynamics and state transitions that define each sequence, thereby validating its applicability to real-world data. Additionally, we explore how different parameters of the HMMs impact the results of the pipeline, to gain an understanding of the sensitivity of the clustering procedure to changes in data complexity. By varying the HMM parameters used to generate the sequences, we were able to create diverse sets of sequences for clustering. The generated sequences were of fixed lengths and sampled multiple times from each HMM to ensure variability within the datasets.

To select the hyperparameters of `DBSCAN`, we employ the same hyperparameter optimisation strategy that we intend to use on real-world datasets.

The first step of the process involves defining distinct HMM models which generate unique sequences of integer values. Next, we follow the process of clustering the sequences similarly to how we use the 2-stage clustering pipeline. We fit a single HMM model using all sequences, then find the distance between the induced transition matrices from each sequence. This step is described in further detail in subsection 3.2.1.

The resulting distance matrix is clustered using the `DBSCAN` algorithm. The `DBSCAN` parameters ϵ and minimum samples are found using hyperparameter optimisation by setting the silhouette value to be maximised during the optimisation.

For each experiment, we generate 50 sequences each of length 10. We used a length of 10 as this length reflects the real-world datasets. A total of 50 sequences were selected as a sample size, for computational efficiency and time constraints. We tested the performance of the HMM by fitting models with varying numbers of states, ranging from 2 to 5, for each experiment to determine the best performance for clustering. After obtaining the distance matrix, we use the hyperparameter optimisation to determine the `DBSCAN` parameters. The search space for the parameters is defined in the table below and will be used for all further experiments unless stated otherwise.

Hyperparameter	**Search Space**
`DBSCAN` (ϵ)	`Uniform` $[0.1, \max(\text{dists})]$
`DBSCAN` (min samples)	`Int Uniform` [2, 50]

Table 3.1: The Hyperopt search space we used for hyperparameter optimisation. The distributions are noted in pseudocode corresponding to the distribution in the Hyperopt package. The `Uniform` distribution is a probability distribution where all the values within the specified range are have the same probability to occur. Whereas the `Int Uniform` distribution is a probability distribution where all integer values within the specified range have equal probability to occur.

The space of parameters was chosen to limit the search within a reasonable range of values of ϵ and minimum samples. The optimisation was run for 600 was enough to converge iterations after which the distance matrix was clustered with the parameters which resulted

in the highest silhouette score. In addition to the silhouette index calculated during the optimisation, we calculate internal measures for the final clustering: Adjusted Rand Index (ARI) [121], Normalized Mutual Information (NMI) [29], Fowlkes-Mallows Index (FMI) [122], high values of these measures reflect that the clustering matches the true labels from which the sequences were generated.

3.4.1 Single HMM Model

For the first experiment, we created a single HMM with 3 hidden states and 2 possible values to emit, with the following start probabilities (π), transition matrix (A) and emission probabilities (B):

$$\pi = \begin{bmatrix} 0.3 & 0.3 & 0.4 \end{bmatrix} \qquad A = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \qquad B = \begin{bmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \\ 0.8 & 0.2 \end{bmatrix}.$$

The sequences are characterised by emitting almost exclusively values based on their respective state value.

Clustering Results

Since the silhouette index is undefined for a single cluster, we assigned a loss value of 0 during the optimisation step whenever a single cluster was detected. This approach was consistently applied in subsequent experiments and throughout the pipeline when tested with real-world datasets. This was a limitation in this specific experiment as the sequences are originating from a single HMM model; therefore, they should all be part of a single cluster. This is discussed in more detail in the result evaluation section. After performing the hyperparameter optimisation to identify the `DBSCAN` parameters, we use internal metrics to measure the quality of the resulting clustering. The hyperparameter optimisation was tested with different numbers of HMM states, the results of this is presented in Table 3.2.

Number of States	ARI	NMI	FMI	ϵ	Min Samples	Silhouette Score
2	0.0	0.0	0.835928	0.005913	27	0.669908
3	0.0	0.0	0.921844	0.000490	19	0.580835
4	0.0	0.0	0.851889	0.008133	29	0.581771
5	0.0	0.0	0.851889	0.007732	27	0.583195

Table 3.2: Cluster evaluation metrics for different numbers of states.

In addition to this, we also present the cluster labels from the clustering results of different numbers of hidden states Figure 3.9.

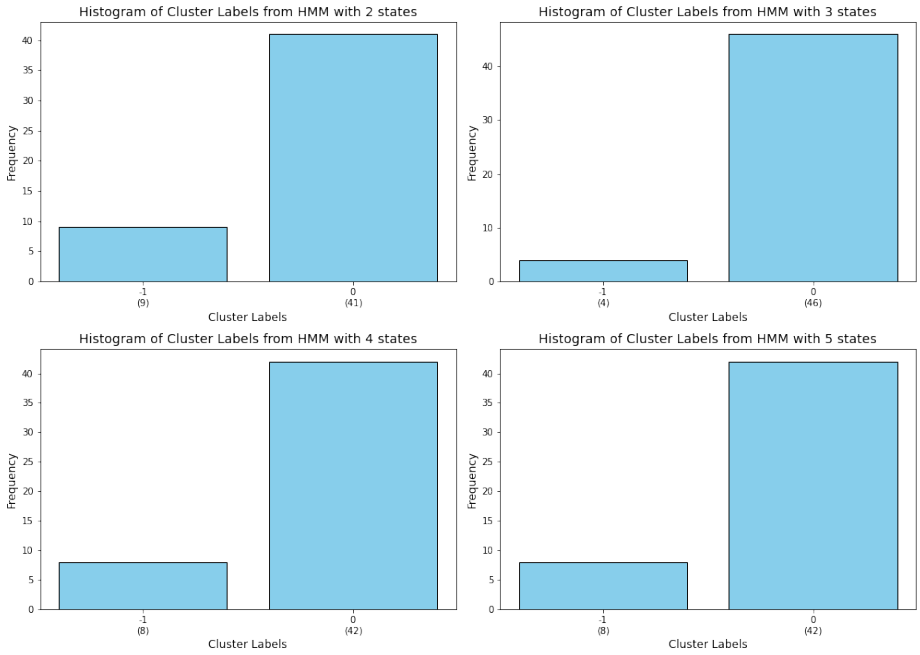

Figure 3.9: Results of a single HMM model.

Discussion of results

When there is only a single cluster present in the true labels, the interpretation of clustering performance metrics like Adjusted Rand Index (ARI), Normalised Mutual Information (NMI), and Fowlkes-Mallows Index (FMI) can be different to when there are more than one cluster. If the true labels consist of a single cluster, ARI is not well-defined for evaluating multiple clusters. Typically, the ARI will be close to zero if the predicted clustering has more than one cluster, indicating no agreement beyond chance. With a single cluster in the true labels, NMI can also be problematic because mutual information requires variability in both clusterings to provide meaningful information. The FMI measure is different to other measures in this case. If the true labels have a single cluster, the precision and recall are defined in relation to how well the predicted clustering identifies the single cluster. If the predicted clustering also assigns all points to one cluster, FMI will be high. This explains the values of ARI and NMI across the different numbers of states for this specific experiment. When looking at the results of the clustering in Table 3.2 the FMI value is consistently high across all numbers of states. Additionally, the cluster label distribution in Figure 3.9 can be seen to largely cluster the sequences into a single cluster with several outliers across the different tests. Despite the limitation of the silhouette index in the hyperparameter optimisation, the results were still mostly accurate, as can be seen in Figure 3.9 as most sequences were correctly identified to be originating from a single cluster.

3.4.2 Experiment with two HMMs

For this experiment, we tested the clustering performance with 2 separate HMM models, each generating distinct sequences. The HMM model parameters are listed below:

$$\pi_1 = \begin{bmatrix} 0.9 & 0.05 & 0.05 \end{bmatrix} \qquad A_1 = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \qquad B_1 = \begin{bmatrix} 0.05 & 0.9 & 0.05 \\ 0.05 & 0.9 & 0.05 \\ 0.05 & 0.9 & 0.05 \end{bmatrix}$$

$$\pi_2 = \begin{bmatrix} 0.05 & 0.9 & 0.05 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 0.05 & 0.9 & 0.05 \\ 0.05 & 0.05 & 0.9 \\ 0.9 & 0.05 & 0.05 \end{bmatrix} \qquad B_2 = \begin{bmatrix} 0.05 & 0.05 & 0.9 \\ 0.05 & 0.05 & 0.9 \\ 0.05 & 0.05 & 0.9 \end{bmatrix}$$

The sequences generated by the 1^{st} HMM model are largely composed of the second value from the emission matrix, while the sequences 2^{nd} HMM are primarily the third value from its respective emission matrix. The HMM models were defined in such a way as to lead to distinct sequences to test the capabilities of the clustering method.

Clustering Results

Results of the hyperparameter optimisation are presented in Table 3.3.

Number of States	ARI	NMI	FMI	ϵ	Min Samples	Silhouette Score
2	0.851885	0.833952	0.922286	0.010611	4	0.976185
3	0.849468	0.805798	0.920973	0.023196	9	0.944519
4	0.959996	0.929125	0.979800	0.045167	14	0.947073
5	0.849576	0.798824	0.921930	0.077352	18	0.925119

Table 3.3: Cluster evaluation metrics for different numbers of states.

The cluster label distributions are presented in Figure 3.10.

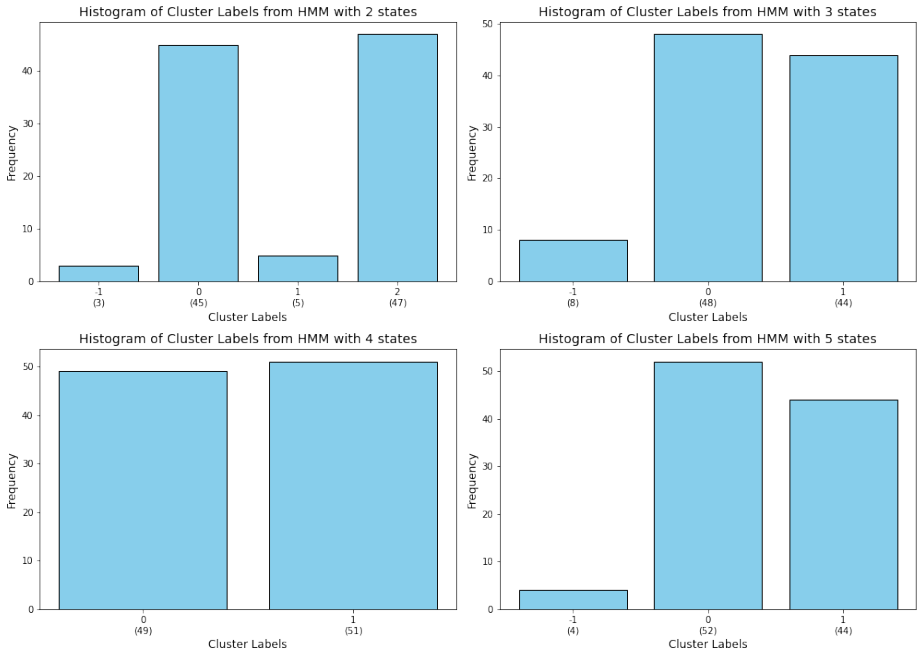

Figure 3.10: Results after cluster of distribution between clusters. All the sequences were correctly separated into 2 clusters.

Discussion of results

The results are summarised in Table 3.3. The Adjusted Rand Index (ARI) measures the similarity between the true cluster assignments and the clusters obtained from the `DBSCAN` clustering. For the configurations tested, ARI values indicate high agreement between the true labels and the clustering results, ranging from 0.849468 to 0.959996. Normalised Mutual Information (NMI) assesses the amount of mutual information shared between the true and predicted clusterings. NMI values range from 0.805798 to 0.929125 across different numbers of states, indicating substantial information sharing. The Fowlkes-Mallows Index (FMI) combines precision and recall to evaluate clustering performance. The consistently high FMI values across different configurations reflect strong precision and recall for the clusters obtained. The Silhouette Score measures the quality of clustering based on how similar each sample is to its own cluster compared to other clusters. Across different configurations, Silhouette Scores range from 0.925119 to 0.976185, indicating well-defined clusters with high cohesion and separation. Looking at the cluster label distributions in Figure 3.10, across all 4 tests, most sequences were separated into 2 separate clusters with only several cases of outliers or supplementary clusters. Overall, the performance of this test was largely successful, as evidenced by high ARI, NMI, FMI, and Silhouette Score values as well as accurate separation of sequences into the correct number of clusters. The choice of 4 states stands out with particularly high scores across multiple metrics, suggesting that this configuration captures the underlying structure of the data effectively.

3.4.3 Experiment with 3 HMMs

To further test the capabilities of the clustering performance, we created 3 distinct HMM models, the matrices for which are listed below:

$$\pi_1 = \begin{bmatrix} 0.8 & 0.1 & 0.1 \end{bmatrix} \qquad A_1 = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \qquad B_1 = \begin{bmatrix} 0.05 & 0.9 & 0.05 \\ 0.05 & 0.9 & 0.05 \\ 0.05 & 0.9 & 0.05 \end{bmatrix}$$

$$\pi_2 = \begin{bmatrix} 0.1 & 0.8 & 0.1 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 0.05 & 0.9 & 0.05 \\ 0.05 & 0.05 & 0.9 \\ 0.9 & 0.05 & 0.05 \end{bmatrix} \qquad B_2 = \begin{bmatrix} 0.05 & 0.05 & 0.9 \\ 0.05 & 0.05 & 0.9 \\ 0.05 & 0.05 & 0.9 \end{bmatrix}$$

$$\pi_3 = \begin{bmatrix} 0.1 & 0.1 & 0.8 \end{bmatrix} \qquad A_3 = \begin{bmatrix} 0.05 & 0.9 & 0.05 \\ 0.05 & 0.05 & 0.9 \\ 0.9 & 0.05 & 0.05 \end{bmatrix} \qquad B_3 = \begin{bmatrix} 0.9 & 0.05 & 0.05 \\ 0.9 & 0.05 & 0.05 \\ 0.9 & 0.05 & 0.05 \end{bmatrix}$$

The HMM models were defined similarly to the ones in the experiment, with 2 HMM models, where each model primarily emits a single value. This was so that all sequences were largely distinct from one another.

3.4.3.1 Clustering Results

The results of the optimisation are listed below in Table 3.4.

Number of States	ARI	NMI	FMI	EPS	Min Samples	Silhouette Score
2	0.362520	0.522375	0.574969	0.004117	2	0.935057
3	0.851544	0.878823	0.900189	0.036672	15	0.852330
4	0.792581	0.822921	0.859860	0.014311	6	0.876012
5	0.979932	0.970191	0.986532	0.084041	33	0.853197

Table 3.4: Cluster evaluation metrics for different numbers of states.

The cluster label distributions are presented in Figure 3.11.

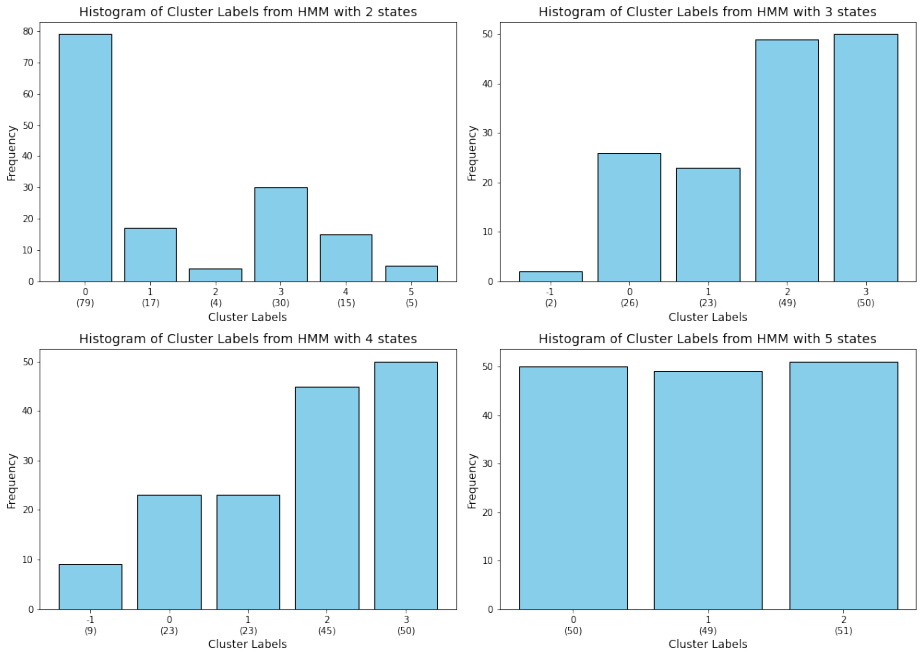

Figure 3.11: Results after cluster of distribution between clusters. All the sequences were correctly separated into 2 clusters.

Discussion of results

The performance of the clustering was evaluated using several metrics: Adjusted Rand Index (ARI), Normalised Mutual Information (NMI), Fowlkes-Mallows Index (FMI), and Best Silhouette Score. The results are summarised in Table Table 3.4. The ARI values show an increasing trend with the number of states. The ARI starts at 0.362520 for 2 states and reaches 0.979932 for 5 states. This indicates that the clustering becomes more consistent with the true labels as the number of states increases. Similarly, the NMI values also demonstrate an improvement with an increasing number of states. For 2 states, the NMI is 0.522375, and it increases to 0.970191 for 5 states. Higher NMI values suggest that the mutual information shared between the predicted clusters and the true labels is higher, indicating better clustering performance. The FMI values follow the same trend as ARI and NMI. Starting at 0.574969 for 2 states, the FMI increases to 0.986532 for 5 states. This index combines the precision and recall of the clustering, and higher values indicate better clustering performance. The results of the cluster analysis indicate that increasing the number of states in the HMM generally improves the clustering performance, as evidenced by higher ARI, NMI, and FMI values. The optimal parameters for the `DBSCAN` algorithm (ϵ and minimum samples) vary with the number of states, reflecting the different clustering resolutions needed for different state configurations. When looking at the cluster label distribution in Figure 3.11, the performance of the test with 2 states was the worst. This is perhaps due to the model with only 2 states not being able to capture the variety of distinct sequences originating from 3 distinct models. On the other hand, as the number of states increases, the performance increases. The tests with 2 and 3 states identified 4 clusters, where 2 clusters were approximately of the

same size as the number of sequences generated from each HMM model (50); however, the other 2 clusters seem to have split the sequences from a single HMM into 2 separate clusters. Overall, the analysis shows robust clustering performance across most tested state numbers, with particularly high silhouette scores indicating well-defined clusters.

3.4.4 Conclusions

In this section, we demonstrated the application of HMM-based clustering on artificially generated datasets created using Hidden Markov Models (HMMs). Our primary objective was to evaluate whether the clustering procedure could accurately identify the number of HMM models from which the artificial sequences were generated. Although the clustering did not recover the underlying latent-state structure of the generative HMMs (i.e., it did not infer the true number of states), the pipeline did reliably group sequences with similar dynamics, even when they were generated by HMMs with different state counts. In this study, the object of interest is the partition of sequences into comparable trajectory clusters and not the identification of HMMs' state numbers. The process began with the definition of distinct HMM models, each generating unique sequences of integer values. We then applied a clustering approach similar to our two-stage clustering pipeline. Initially, a single HMM model was fitted using all sequences. Subsequently, we calculated the distances between the induced transition matrices of each sequence. This distance matrix was then clustered using the `DBSCAN` algorithm, with the parameters ϵ and minimum samples optimised through hyperparameter tuning to maximise the silhouette value. By using multiple HMMs to generate artificial sequences, we simulated systems with varying temporal dependencies. Each HMM was defined by specific start probabilities, transition matrices, and emission probabilities, allowing us to produce diverse sets of sequences for clustering. The sequences were of fixed lengths and were sampled multiple times from each HMM to ensure variability within the datasets. The clustering results confirmed that our methodology could successfully identify the distinct HMM models from which the sequences were derived. This validation of Stage 2 of our clustering pipeline underscores its ability to discern the number of underlying HMM models, thereby demonstrating its effectiveness in capturing the inherent structure and temporal dependencies within the data.

Chapter 4

Optimising Clustering Analysis: Grouping and Visualising Equivalent Solutions

During hyperparameter optimisation of clustering analysis, it is common to encounter multiple equivalent solutions, especially in scenarios where the optimisation landscape exhibits symmetry or redundancy. In this context, equivalent solutions refer to distinct sets of hyperparameters that produce nearly identical clustering outputs. These equivalent solutions can create challenges in hyperparameter tuning, leading to ambiguity in selecting the optimal clustering solution.

In this chapter, we introduce a comprehensive methodology for grouping and analysing clustering solutions based on their similarity. This approach is supported by a graphical user interface (GUI) designed for visualising and interpreting these relationships. The chapter addresses the challenges of evaluating and distinguishing between numerous clustering solutions that arise during hyperparameter optimisation, where multiple equivalent solutions complicate the selection process.

To address these challenges, we propose a similarity-based graph approach for comparing and categorising clustering solutions. In this graph, each node represents a unique clustering solution, while edges indicate the similarity between pairs of solutions. This representation allows us to systematically identify distinct solutions while filtering out redundant or equivalent ones, simplifying the analysis of optimisation results. We also introduce prototype solutions, which act as representatives of similar groups of clustering solutions, thus reducing the number of solutions to analyse without losing meaningful diversity. Figure 4.1 presents an overview of the process for obtaining these prototype solutions.

The chapter also details the development of a GUI designed for exploring and analysing these similarity graphs. This tool provides a user-friendly interface, allowing researchers to interactively visualise relationships between clustering solutions and enhance interpretability. Additionally, we introduce the use of Sankey diagrams to illustrate how different clustering solutions relate to one another, providing deeper insights into the equivalence and distinctions between solutions.

Figure 4.1: General overview of how to obtain a subset of prototype solutions from the original dataset. A dataset is first clustered using an arbitrary clustering pipeline, next the clustering solutions are analysed by constructing a graph using distances between solutions as. Prototype solutions are extracted from the Solution graph and represent unique ways of partitioning the dataset.

4.1 Graph of Clustering Solutions

This section introduces the graph of clustering solutions and the process of extracting unique solutions from the graph. Starting with a large space of possible clustering solutions, our goal is to extract distinct solutions by measuring similarity between them. Measuring the similarity between clustering solutions enables us to visualise their relationships in a two-dimensional graph representation. Using this graph, we set a similarity threshold to identify clusters of similar solutions, from which we select a single representative solution—referred to as the prototype solution. These prototype solutions provide a simplified yet representative overview of the entire solution space.

Given a set of clustering solutions $\mathbf{F} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_u\}$, we aim to compute the pairwise similarity between each pair of solutions and construct a graph where each node represents a clustering solution and each edge represents the similarity between two solutions. The steps of this are described as follows.

The similarity measure $S(\mathbf{C}_i, \mathbf{C}_j)$ quantifies how similar two clustering solutions $\mathbf{C}_i$ and $\mathbf{C}_j$ are to one another. Specifically we use the **element-centric clustering comparison** [123] a similarity function $S : \mathbf{C} \times \mathbf{C} \to \mathbb{R}$ such that $S(C_i, C_j)$ quantifies the similarity between clustering solutions $\mathbf{C}_i$ and $\mathbf{C}_j$.
Given two clusterings $\mathbf{C}_1$ and $\mathbf{C}_2$ of the same n elements, we follow [123]. For a clustering $\mathbf{C}$, let $B^{(\mathbf{C})} \in \mathbb{R}^{n \times k}$ be the element–cluster membership matrix (hard or soft), and $S = \mathrm{diag}(s_1, \ldots, s_k)$ with $s_c = \sum_{u=1}^{n} B_{uc}^{(\mathbf{C})}$. Projecting the bipartite affiliation graph yields the cluster–induced element graph with

$$W^{(\mathbf{C})} = B^{(\mathbf{C})}\, S^{-1} \left(B^{(\mathbf{C})}\right)^{\top}.$$

Let $P^{(\mathbf{C})} = D^{-1} W^{(\mathbf{C})}$ be the row–stochastic transition matrix (D is the degree diagonal). For each element u, the personalised PageRank (PPR) vector is the solution of

$$\boldsymbol{\pi}_u^{(\mathbf{C})} \;=\; \alpha\, \mathbf{e}_u \;+\; (1-\alpha)\left(P^{(\mathbf{C})}\right)^{\top} \boldsymbol{\pi}_u^{(\mathbf{C})}, \quad \alpha \in (0,1),$$

which serves as the element-centric association profile for element u. The per–element agreement between $\mathbf{C}_1$ and $\mathbf{C}_2$ is the corrected L_1 agreement between PPR distributions,

$$s_u \;=\; 1 - \widetilde{L}_1\big(\boldsymbol{\pi}_u^{(\mathbf{C}_1)},\; \boldsymbol{\pi}_u^{(\mathbf{C}_2)}\big),$$

and the element–centric similarity is

$$S(\mathbf{C}_1, \mathbf{C}_2) \;=\; \frac{1}{n}\sum_{u=1}^{n} s_u.$$

This formulation was used as it is label–free, size–aware, and accommodates overlapping and hierarchical clusterings [123].

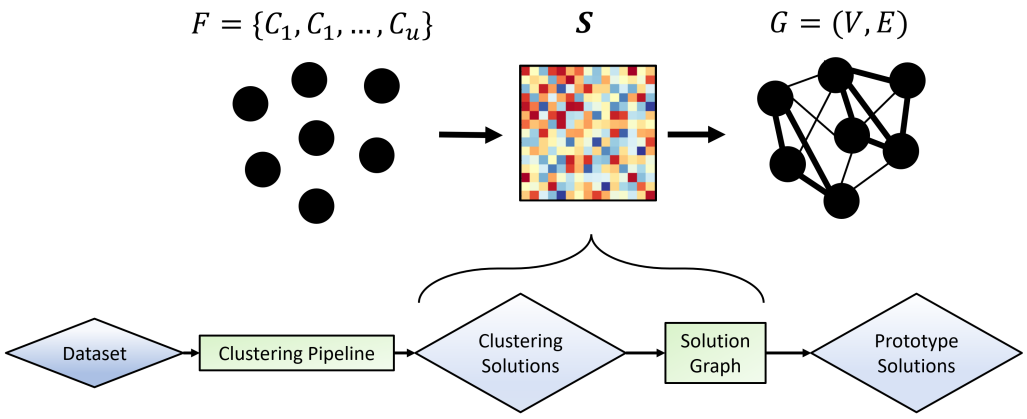

Figure 4.2: The dataset is first clustered using an arbitrary clustering pipeline, and the resulting clustering solutions are analysed by constructing a similarity graph. Prototype solutions are then extracted from the solution graph, representing unique ways of partitioning the dataset.

$$S : \mathbf{C} \times \mathbf{C} \to \mathbb{R}$$
$$(C_i, C_j) \mapsto S(\mathbf{C}_i, \mathbf{C}_j)$$

After computing the similarities between all clustering solutions in $\mathbf{F}$, we use $\mathbf{S}$ to denote the similarity matrix between the solutions. Next, to construct an undirected graph $G = (V, E)$ where each node $v_i \in V$ represents to a clustering solution $\mathbf{C}_i \in \mathbf{F}$. The edge set E is defined such that an edge $e_{ij} \in E$ exists between nodes v_i and v_j with weight w_{ij} corresponding to the similarity $S(\mathbf{C}_i, \mathbf{C}_j)$ between the 2 solutions.

$$G = (V, E)$$
$$V = \{v_1, v_2, \ldots, v_v\}$$
$$E = \{e_{ij} \mid v_i, v_j \in V \text{ and } e_{ij} = (v_i, v_j, w_{ij})\}$$
$$w_{ij} = S(\mathbf{C}_i, \mathbf{C}_j)$$

The graph G corresponds to all clustering solutions in the dataset $\mathbf{F}$ and how similar these solutions are to each other. However, depending on the size of $\mathbf{F}$, the graph G can still be difficult to interpret. The graph of solutions, as shown in Figure 4.2, can be inspected to relate the width of the edges to the magnitude of the similarity between specific connections;

however, more insight into the clustering solutions can be extracted.

To identify unique groups of clustering solutions from G we introduce a method of retaining connections between solutions that are similar only within a specified threshold. Introducing a similarity threshold removes connections between solutions that barely share any clustering information with each other. After this threshold is applied, the previous fully connected graph is split into several connected components or, as we define them, **clusters of solutions**. The clusters of solutions are connected nodes with edge weights above the specified threshold and represent groups of solutions that share a significant amount of overlap between the memberships of the elements in the clustering. If there are only a limited number of unique ways of clustering the dataset present in F, these unique clusterings will be represented by the clusters of solutions after setting this threshold value. Details of this process are specified next.

Let θ be a **similarity threshold**. To find clusters within the graph G, we apply a similarity threshold θ to the edge weights S. Retaining only the edges e_{ij} with weights $w_{ij} \geq \theta$. Form a subgraph $G' = (V, E')$ where:

$$E' = \{e_{ij} \mid w_{ij} \geq \theta\}$$

The connected components in the new graph G' can be identified next. Each connected component $\mathcal{C}_i$ represents a cluster of similar clustering solutions.

In summary, the process can be encapsulated as:

$$\mathbf{F} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_u\}$$

$$S : \mathbf{C} \times \mathbf{C} \to \mathbb{R}$$

$$G = (V, E) \text{ with } V = \{v_1, v_2, \ldots, v_v\} \text{ and } E = \{e_{ij} = (v_i, v_j, w_{ij}) \mid w_{ij} = S(\mathbf{C}_i, \mathbf{C}_j)\}$$

$$G' = (V, E') \text{ with } E' = \{e_{ij} \mid w_{ij} \geq \theta\}$$

After this process, the resulting groups of solutions $\mathcal{C}_i$ are the connected components of G'. This is represented in a diagram in Figure 4.3.

Using this methodology, we have identified groups of distinct clustering results from a large set of clustering solutions. This process is visualised in Figure 4.3. This method of identifying clusters of solutions is further expanded on in the next section by only selecting a single solution as the representative **prototype** solution.

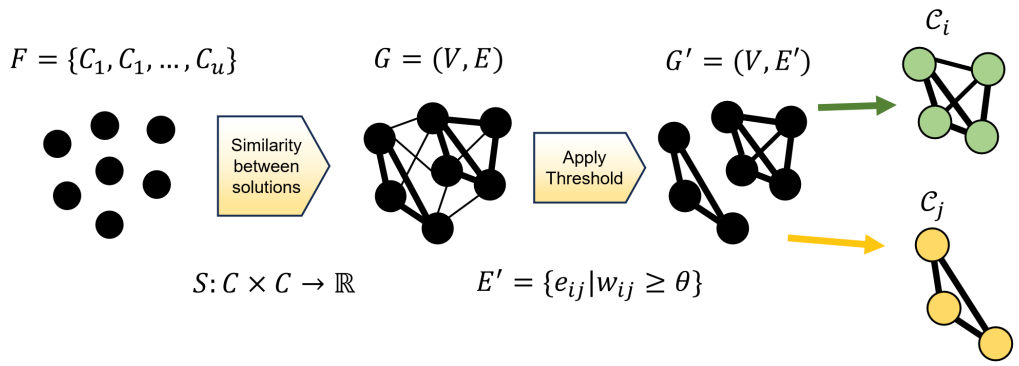

Figure 4.3: Diagram of the identifying prototype clustering solutions.

4.2 Prototype Solutions

A longstanding challenge in cluster analysis is selecting the most appropriate clustering solution, as different disciplines and applications may require alternative solutions [124]. In the previous section, we introduced a methodology for constructing a graph where nodes correspond to clustering solutions and edges correspond to the similarity between them. By applying a threshold to this graph, we can partition it into clusters of solutions, each representing a distinct way of clustering the same dataset. This approach offers a clear overview of the number of unique clustering solutions present within the solution set and the frequency of their occurrence.

In this section, we introduce the concept of a **prototype** solution, a single clustering solution from each group of solutions identified from the graph. The prototype solution represents a unique way of clustering the dataset. This allows for the condensing of a large set of possible clustering solutions with many overlapping and similar solutions into several prototype solutions, each corresponding to a unique partition of the dataset. This prototype solution is determined by identifying the solution with the highest internal or external measure. The details of this are described below.

Let $\mathcal{C}_i$ be the connected components of the solution graph G' after applying a threshold of similarity, described in detail in the previous section. The solution with the highest internal or external validation measure in $\mathcal{C}$ is the prototype solution P. This can be formalised as:

$$P = \arg\max_{C \in \mathcal{C}} s_{\text{val}}(\mathbf{C}, \mathcal{C}), \tag{4.1}$$

where s_{val} is the validation measure of solution $\mathbf{C}_j$ in cluster of solutions $\mathcal{C}_i$. This is visualised in Figure 4.4.

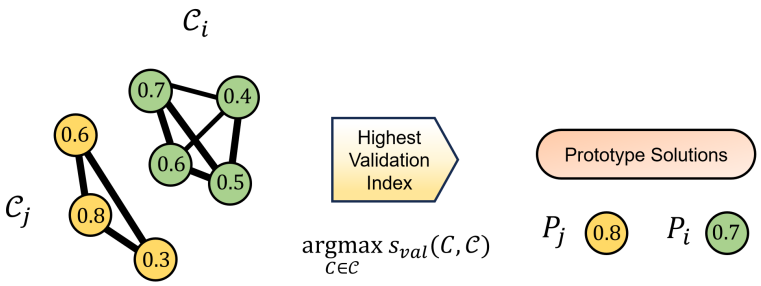

Figure 4.4: Diagram of the selecting prototype solutions from clusters of solutions $\mathcal{C}_i$ and $\mathcal{C}_j$ from Figure 4.3. The values in the centre of each node are the validation indices for the specific solution.

Many clustering results are equivalent across a large space of possible clustering solutions. Our goal in this section is to remove redundant solutions and only select distinct and unique solutions. We define prototype solutions to be these unique solutions, which represent unique partitions. A further, more intuitive explanation of this is depicted in Figure 4.5. In the figure, 4 different clustering solutions of the same dataset are shown. Some of these clustering solutions are more similar to one another, where the differences between solutions are minimal; we refer to these solutions as equivalent or redundant. From the 4 solutions, it can be seen that the space of solutions can be cut in half, as only 2 solutions are actually distinct from one another. In cases where some solutions are not found to be similar to any other solutions in the dataset, these are also considered prototype solutions, as they represent a unique partition of the dataset.

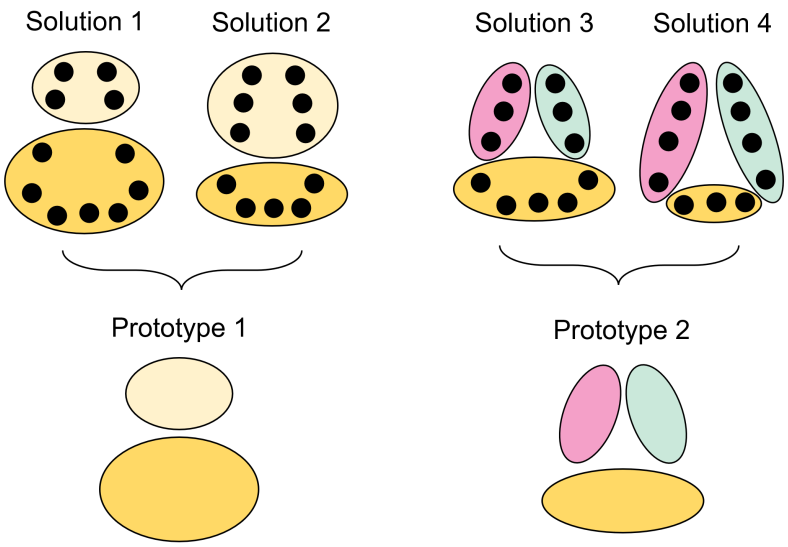

Figure 4.5: An example showing how 4 solutions can be simplified into 2 unique solutions. Solutions 1 and 2 both have 2 clusters with only 2 elements changing memberships between the solutions. Solutions 3 and 4, on the other hand, have 3 clusters and would intuitively be considered different from solutions 1 and 2. Out of the 4 possible clustering solutions, we can identify solutions 1 and 3 as the prototype solutions for this specific example, as they represent unique ways of partitioning the dataset.

4.2.1 Visualising Prototype and Solution Clusters

The most common technique of visualising clustering solutions is by colour coding points in a 2-d space to represent different cluster memberships. An example of this is presented in Figure 4.6.

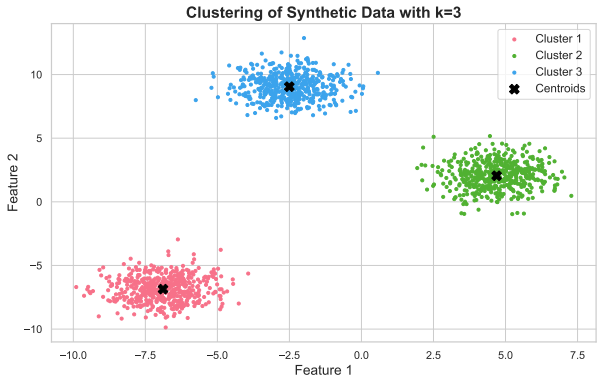

Figure 4.6: Clusters represented as clouds in a 2-d space.

This visualisation is a useful technique; however, when several possible clustering solutions

exist, presenting these becomes difficult. Presenting several plots in the style of Figure 4.6 with different memberships can become overwhelming with increasing numbers of solutions. Additionally, differences between element memberships across clustering solutions are not as easy to identify. To address this, we introduce a method of visualising clustering solutions using Sankey Diagrams. Sankey Diagrams were introduced in Chapter 3 to present the sequence clusters.

Here, we introduce the use of Sankey diagrams to visualise the flow of element memberships across different clustering solutions. The nodes in these Sankey diagrams are scaled based on the size of the clusters, and the thickness of the connections in the Sankey plots reflects the number of samples which change cluster memberships across the different solutions. An example case of this with 2 clustering solutions from Figure 4.5 is shown in Figure 4.7.

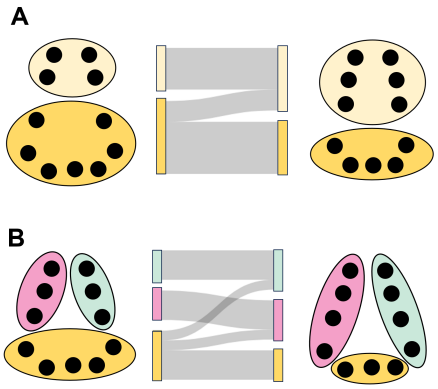

Figure 4.7: Example of using sankey diagrams to represent changes in clustering solutions. In section A) 2 clusters change in size in different solutions. In section B) there are 3 clusters in each solution and 2 objects shift to different clusters.

In Figure 4.7 part A) shows 2 solutions, each with 2 clusters. The solutions only differ by 2 elements, represented by the flow from the left to the right side. Part B) of the Figure 4.7 has 3 clusters, where 2 elements of the yellow cluster on the left-hand side move to the pink and mint colored clusters. Instead of having to search for differences between the solutions, the Sankey diagram shows what proportion of elements moved to another cluster.

4.3 Application to Simulated Dataset

To test the approach introduced in the previous sections, we unified the solution graph and selection of prototype solutions in a single example case. We present a scenario with 9 simulated clustering solutions of the same dataset and test our methodology of constructing a graph using the element-centric similarity score as the edges and clustering solutions as nodes. We then introduce a threshold value to find connected components of clusters of similar solutions. The aim of this is to show a use case of identifying unique clustering solutions from a dataset of several clustering solutions.

The set of 9 clustering solutions is presented in Figure 4.8, with 14 elements ordered in a clockwork pattern and colored based on cluster memberships. We construct a convex hull

to represent the shape of the cluster formed by connecting elements in the same cluster, providing a minimal bounding region that encompasses all the points.

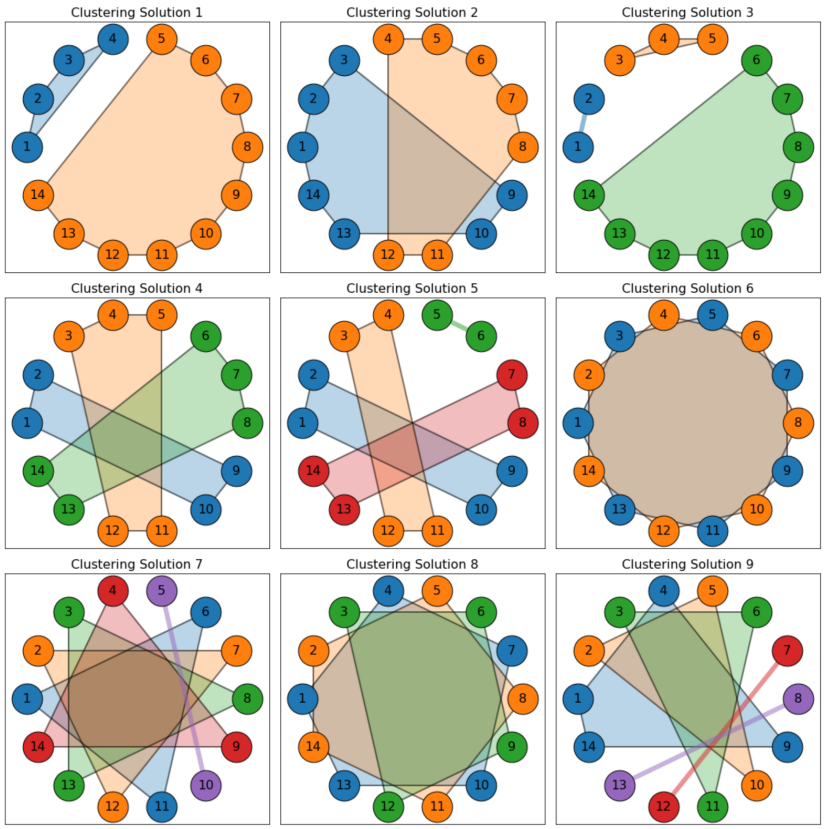

Figure 4.8: Different clustering solutions.

To illustrate a wide range of clustering examples, the partitions in Figure 4.8 vary from 2 clusters to 5 clusters and memberships are shifted across the different clustering solutions.

To visualise these solutions in a graph as described in Section 4.2, the similarity matrix is calculated between all 9 solutions and used to construct a weighted graph displayed in Figure 4.9. The similarity between each solution is displayed atop the edges in the graph. The graph is fully connected as each solution has a similarity to another solution.

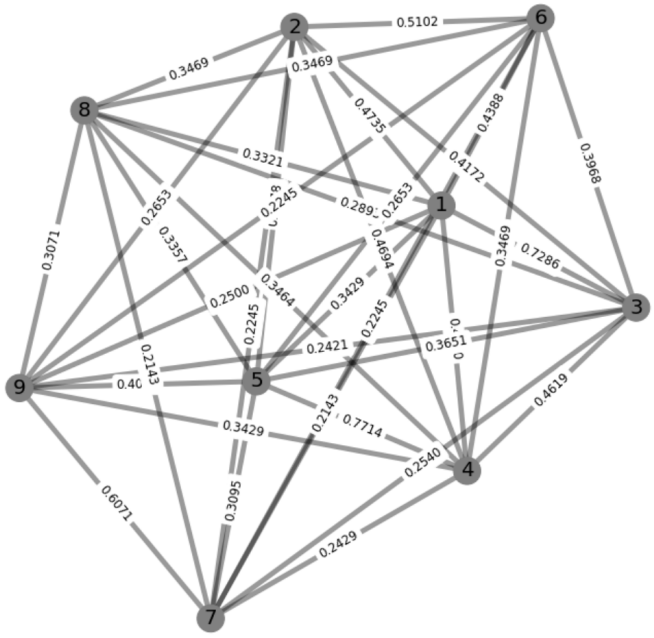

Figure 4.9: Fully connected graph of solutions.

This graph representation can be useful to inspect the similarity between solutions. To further the understanding of how the solutions are related, it is useful to filter more and less connected solutions and group the more connected solutions together. For this purpose, we introduce the threshold similarity value and remove connections that fall below the threshold.

To understand how the graph changes with different threshold values, the threshold is increased from 0 to 1 in increments of 0.01. The solution graph with a threshold of 0.35 is presented in Figure 4.10. Edges in the graph below the threshold value are plotted in a transparent grey, while the connections above the threshold maintain the original grey from the fully connected graph. The threshold of 0.35 was chosen as this was the first value where a solution was separated from the rest of the graph. Solution 8 can be seen as the most distinct from the rest of the solution set. This can be seen in Figure 4.8, where the cluster memberships of elements in Solution 8 are not shared with any other clustering solution.

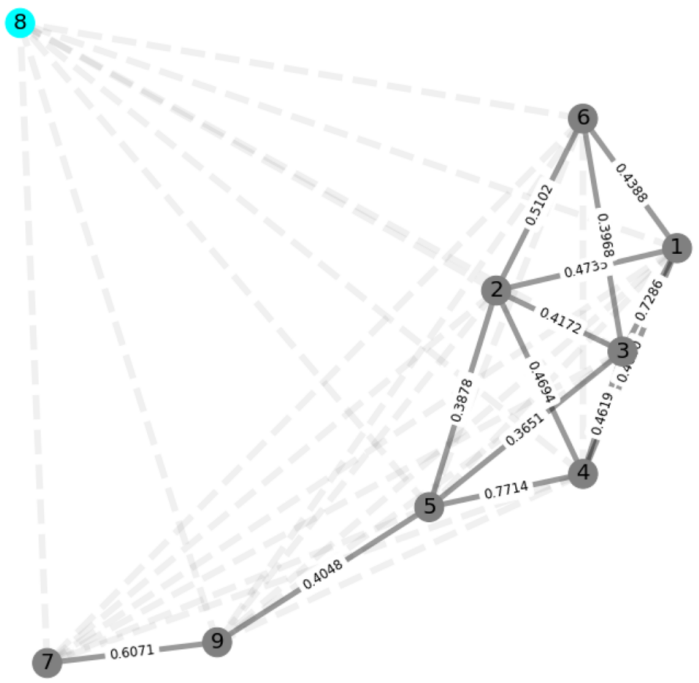

Figure 4.10: Graph of solutions with a similarity threshold of 0.35. The connections below the threshold have been changed to a transparent grey colour.

To illustrate Solution 8 and how different it is from the rest of the solution set, a sankey diagram of clustering memberships is shown in Figure 4.11. Solution 8 is the least similar to the other solutions in the result dataset, as it has a very unique cluster composition, and this partition of elements does not resemble the other solutions. This does not imply that solution 8 is not worth inspecting or analysing further, only that compared to other solutions among the results, it stands out as unique.

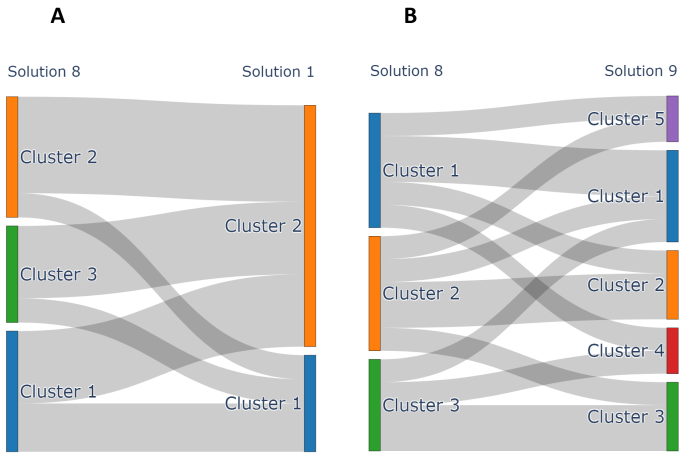

Figure 4.11: Sankey diagrams for Solution 8 with Solutions 1 and 9. Solution 8 was found to be the least similar to any of the other clustering results in the dataset. This image illustrates this using the previously introduced Sankey visualisation for clustering solutions. Memberships from solution 8 share little similarity to the 2 solutions presented here.

If the similarity threshold is further increased to 0.5, the solutions form 4 clusters of solutions, displayed in Figure 4.12. The only solution not clustered with any other solution is Solution 8.

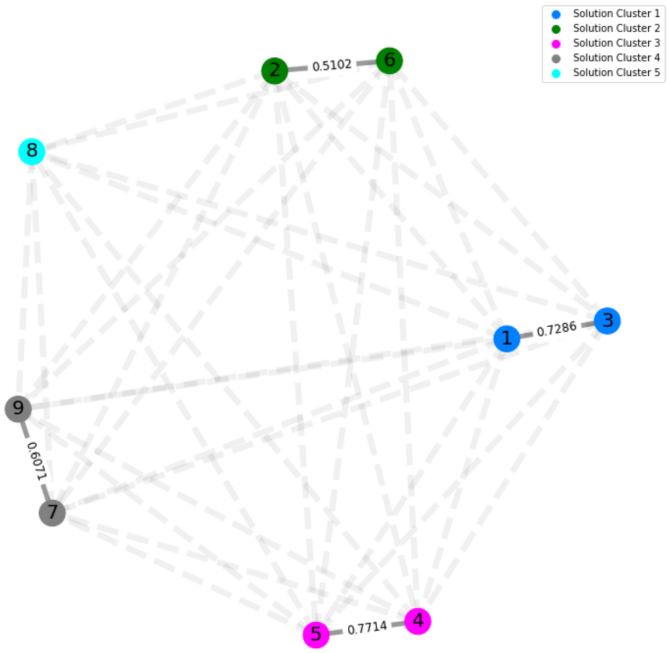

Figure 4.12: Graph of solutions with a similarity threshold of 0.5.

To understand how the solution clusters are related, we use the Sankey cluster diagrams. The first step is to investigate the **intra-cluster** relationship between solutions. In Figure 4.13 and Figure 4.14, the sankey diagrams are displayed from the Solution clusters of Figure 4.12. The figures highlight how the memberships of solutions in the same solution cluster are largely overlapping.

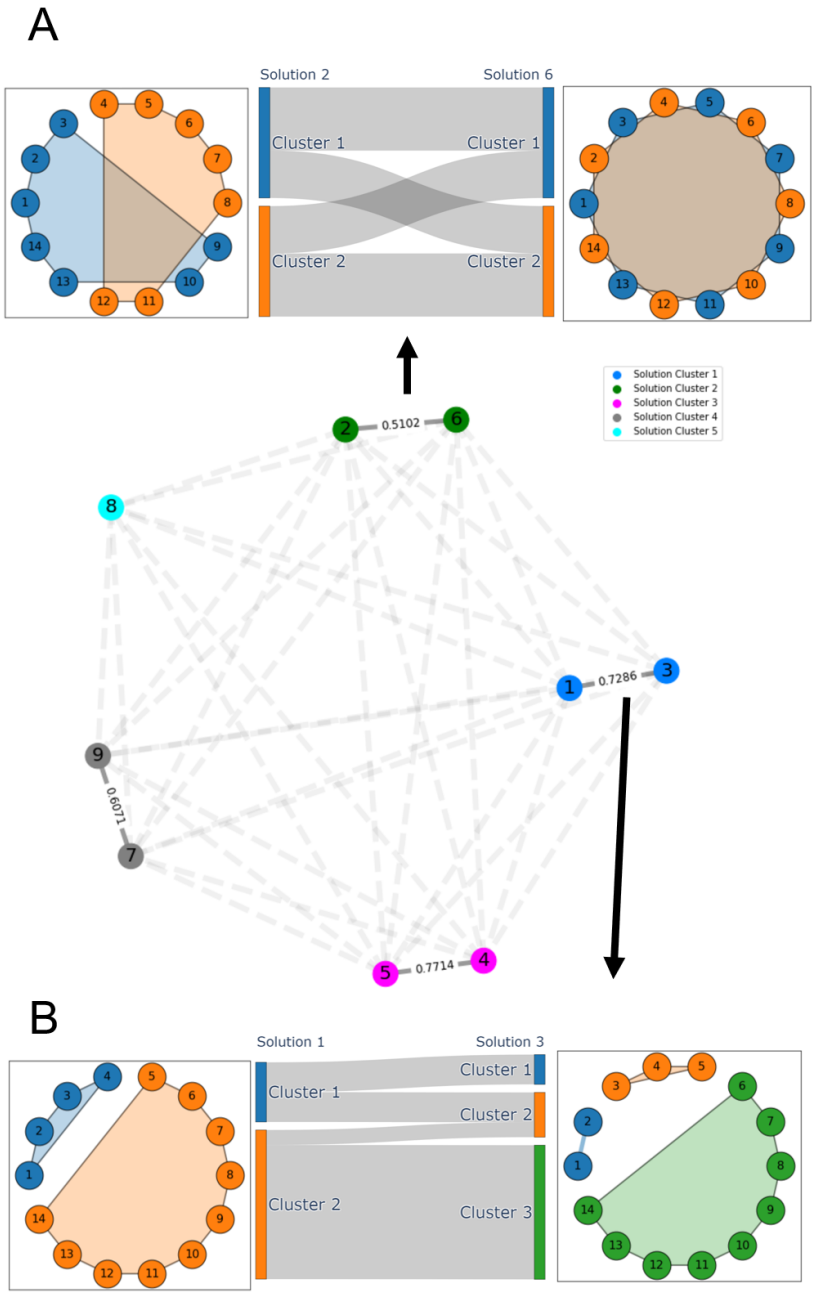

Figure 4.13: Connected components with the respective sankey diagrams.

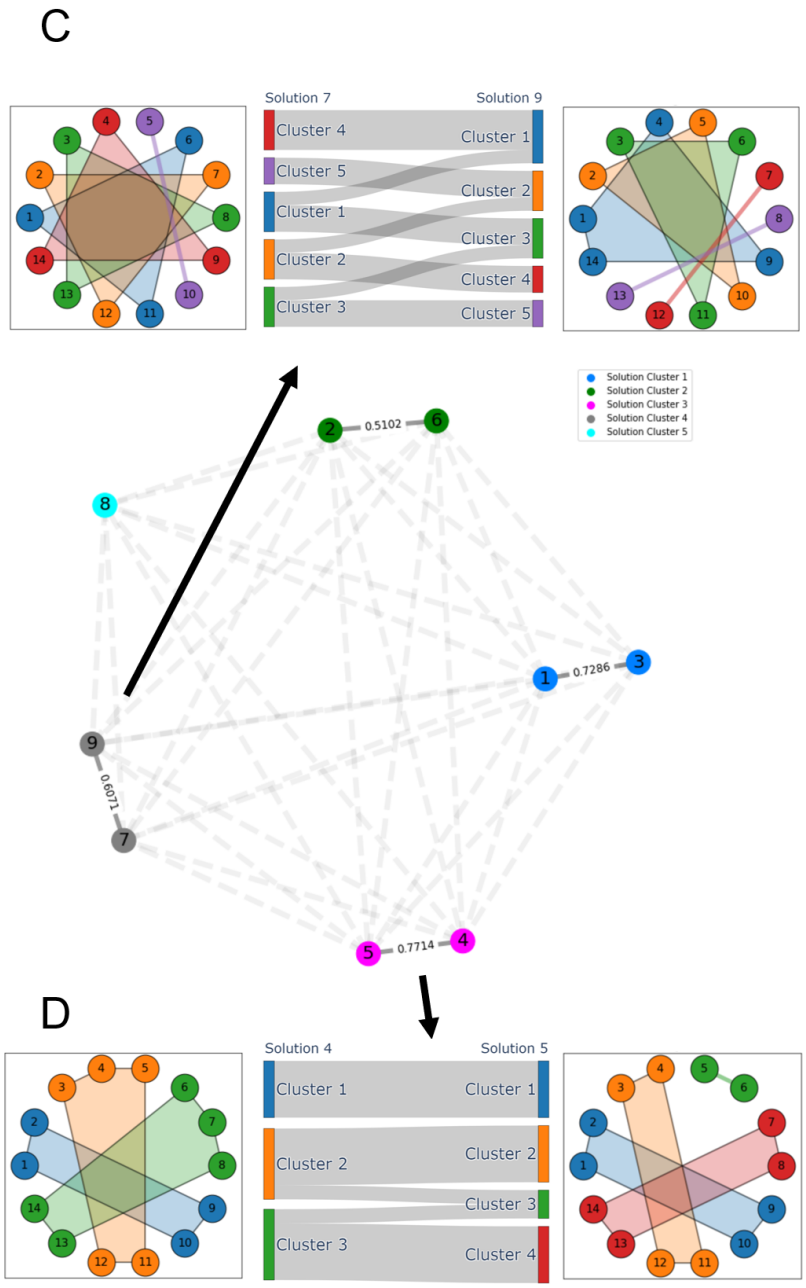

Figure 4.14: Connected components with the respective sankey diagrams.

Part B) of Figure 4.13 shows solution cluster 2, comprised of solutions 1 and 3. Both solutions have grouped elements 6 through 14 into the same cluster, although element 5 in Solution 1 is part of a different cluster. The large proportion of elements belonging to the same cluster across different solution is largely the reason why the element-centric similarity measure is high between these 2 specific solutions. Similar trends of elements belonging to the same clusters across different partitions are displayed in part D) of Figure 4.14.

In addition to intra-cluster sankey diagrams for solutions, we also present sankey diagrams for **inter-cluster** solutions in Figure 4.15. A single solution from each solution cluster is selected along with Solution 8 as the unique solutions from the entire dataset. This is equivalent to the methodology of selecting a prototype solution based on the highest validation index, even though in this simulated example, we have made the solution selection arbitrarily. The sankey diagram in Figure 4.15 is a representation of the 5 unique partitions from the 9 possible clustering solutions.

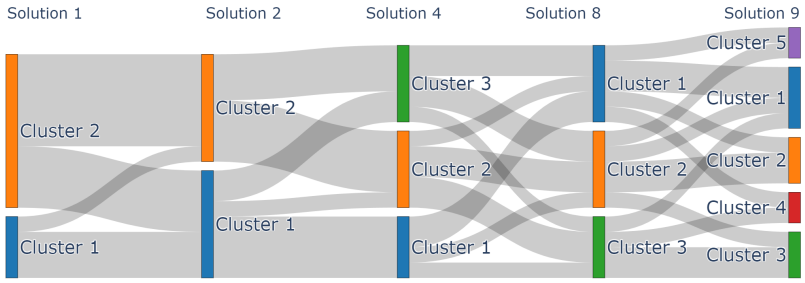

Figure 4.15: Connected components with the respective sankey diagrams.

In this section, we have described a series of steps taken to analyse a set of possible clustering results for the same dataset or partitions. From a dataset of 9 partitions, we first constructed a graph based on the similarity between all solutions, and later created groups of similar solutions. This methodology has provided a way of reducing the space of possible solutions and a visualisation tool for understanding relationships between different solutions.

To use this methodology with other datasets and produce sankey diagrams for intra and inter-cluster comparisons, we developed a Graphical User Interface (GUI).

4.4 ClusterView: A GUI for Exploring Clustering Solutions

To effectively analyse and interpret clustering solution results, it is essential to have a user-friendly interface that integrates all the methods and plotting techniques introduced in the previous sections. The goal of this section is to discuss the development of such an interface—a unified graphical user interface (GUI)—designed to be utilised with other datasets of clustering solutions.

A broad literature supports visual interpretation and comparison of clustering. Surveys cover interactive and graph-centric approaches [125, 126]. Core techniques include dynamic evaluation via parallel cluster views and interactive dendrograms [127], hybrids such as dendrogram-table views [128], multilevel navigation (SnakeTrees) [129], heatmaps with parallel coordinates [130], and force-directed layouts for multi-component graphs [131]. These ideas are integrated into systems like XCluSim [132], Clustervision [133], Clustrophile 2 [134], and VICTOR [135], spanning applications from bioinformatics and medical records to large-scale networks [134, 136, 137]. Despite this progress, several challenges remain. One common limitation across systems is the difficulty of comparing large collections of clustering solutions, particularly when only a few are meaningfully distinct and many are redundant. While tools like Clustrophile 2 and VICTOR allow for user-driven exploration and ranking of results, they

often rely on limited visual summaries or require iterative manual inspection. This creates a need for more systematic techniques to structure the space of clustering solutions, reduce redundancy, and highlight patterns across different solutions.

The creation of a GUI serves several critical purposes in the context of clustering solution analysis using the graph of solutions methodology. Firstly, clustering analysis often involves working with large, complex datasets where manual inspection and analysis of individual solutions can be time-consuming and prone to errors. It is often necessary to filter the solutions to only select a specific subset based on certain criteria. A well-designed GUI can streamline this process by providing a cohesive platform that allows users to efficiently explore and interpret the results.

CluterView is developed within the Jupyter Notebook platform, a widely used environment for data analysis, which offers an interactive interface for writing and running code. Since Jupyter Notebooks are a common workspace for data scientists and researchers, incorporating the GUI into this platform allows users to seamlessly transition from data analysis to visualisation and interpretation without leaving the environment they are accustomed to. This integration enhances workflow efficiency and makes the tool more accessible to users already familiar with Jupyter.

4.4.1 User Requirements and Features

- **Selection of Solution Dataset:** The GUI should allow users to easily select the dataset containing the clustering solutions they wish to analyse. This feature is crucial for handling multiple datasets and enabling comparisons between different clustering methods or parameter settings.

- **Threshold Selection:** Manually changing the similarity threshold is useful as certain datasets may present different patterns of cluster similarity. Having the ability to adjust this value can give insight into the partitioning of the graph.

- **Parameter Selection for Data filtering:** The ability to filter and select subsets of the dataset based on specific parameters is essential. This feature allows users to focus on particular aspects of the data, such as specific clustering solutions or ranges of similarity scores, making the analysis more targeted and manageable.

- **Plots of Selected Parameters:** Visualisation is a key component of clustering analysis. The GUI should provide options to generate plots of the selected parameters, enabling users to quickly identify patterns, outliers, or other notable features within the data.

- **Graph of Solutions:** A visual representation of the clustering solutions in the form of a graph helps users understand the relationships between different clusters. This feature is for visualising how solutions are connected based on similarity measures.

The development of a GUI for clustering solution analysis is driven by the need for a user-friendly, efficient, and integrated tool that can handle the complexities of large datasets. By incorporating all necessary features within the familiar Jupyter Notebook environment, this GUI will significantly enhance the ability of researchers and data scientists to interpret,

understand, and visualise clustering results, ultimately leading to more informed decision-making and deeper insights into the data.

4.4.2 Design and Functionality of the GUI

To address the challenges inherent in evaluating and comparing clustering solutions, we have developed a specialised graphical user interface (GUI). This section presents the background, design, and functionality of the GUI, which integrates advanced visualisation techniques, similarity measures, and quality metrics to provide a comprehensive tool for clustering evaluation.

Our approach to solving this problem involved the development of a GUI that transforms multiple clustering solutions into an interactive graph-based representation. In this graph, each node represents a distinct clustering solution, while edges between nodes represent the similarity between these solutions. This visualisation not only aids in comparing different solutions but also enhances the interpretability of the results, making it easier for users to identify patterns, trends, and insights within the data.

The GUI consists of several interactive elements designed to facilitate the analysis of clustering solutions. The interface is divided into segments that correspond to different stages of the analysis process, each providing specific functionalities and addressing the user's requirements.

- **Dataset Selection and Parameter Adjustment:** As shown in Figure 4.16, the first segment of the GUI allows users to select the dataset they wish to analyse. Section A of the interface provides a dropdown menu for dataset selection, while Section B includes sliders and input fields for adjusting various clustering parameters. Users can filter solutions based on these parameters, ensuring that only relevant solutions are included in the analysis. Section C provides options for plotting the graph and saving the images directly to the local machine.

- **Clustering Solution Analysis:** The second segment, illustrated in Figure 4.17, focuses on the analysis of the selected clustering solutions. Section A displays the number of solutions that remain after applying the filters, providing an overview of the dataset's characteristics. Histogram plots in this segment (shown in red) depict the distribution of various parameters such as Event Group (EG) values, outlier ranges, and the number of hidden Markov model (HMM) states. These visualisations help users to quickly assess the spread and characteristics of the solutions under consideration.

- **Graph Visualization and Component Analysis:** The final segment of the GUI, presented in Figure 4.18, showcases the graph of clustering solutions. In this visualisation, each node represents a clustering solution, and the edges between nodes indicate the similarity between the solutions. Nodes are colour-coded based on their connected components, making it easier to identify groups of similar solutions. Section A highlights the previously selected parameters of the solution dataset, while Section B provides a visual representation which allows users to intuitively understand the relationships between different clustering solutions and to identify clusters of solutions that share significant similarities.

- **Producing Sankey diagrams for Comparisons of Solutions:** In addition to producing plots for the original clustering solutions, Sankey diagrams for comparing connected clustering solutions (as discussed in Section 4.3) are also produced. In Figure 4.19, Sankey diagrams are produced for solutions within the same connected component, while in Figure 4.20 the diagrams are produced to compare solutions from separate connected components.

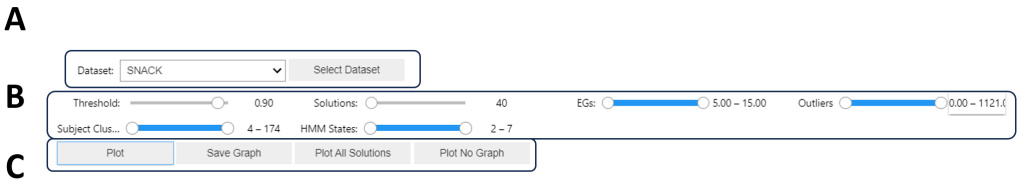

Figure 4.16: Parameter selection section for the Graph GUI.

The GUI is designed to be a comprehensive and intuitive tool for clustering solution analysis, offering several key features that enhance its usability and effectiveness. One of the primary capabilities of the GUI is its interactive dataset and parameter selection. This feature allows users to easily choose different datasets and adjust key parameters that influence the clustering process. Such flexibility is essential for tailoring the analysis to specific datasets and research questions, enabling a more focused and effective exploration of clustering solutions.

The core functionality of the GUI is its graph-based visualisation of clustering solutions. This feature transforms clustering solutions into a network of nodes and edges, making it easier for users to compare and contrast different solutions. By visualising the similarities and groupings within the dataset, users can more readily identify which solutions are most similar to each other and how they are grouped.

Furthermore, the GUI enhances the understanding of solution similarity by representing clustering solutions as nodes connected by similarity-based edges. This intuitive representation helps users grasp the relationships between different solutions, making it particularly useful for identifying clusters of solutions that may represent alternative interpretations of the same data.

Lastly, the GUI automatically divides the graph into components, where each component contains solutions that are more similar to each other than to those in other components. This component-based solution analysis facilitates the identification of groups of solutions that share significant similarities, offering deeper insights into the structure of the data.

In conclusion, the GUI developed in this project serves as a powerful tool for clustering solution analysis. It offers a user-friendly platform that integrates advanced visualisation techniques with robust analytical capabilities. By providing a visual, interactive means to explore and compare clustering solutions, this GUI enhances the interpretability of clustering results and supports more informed decision-making in data analysis.

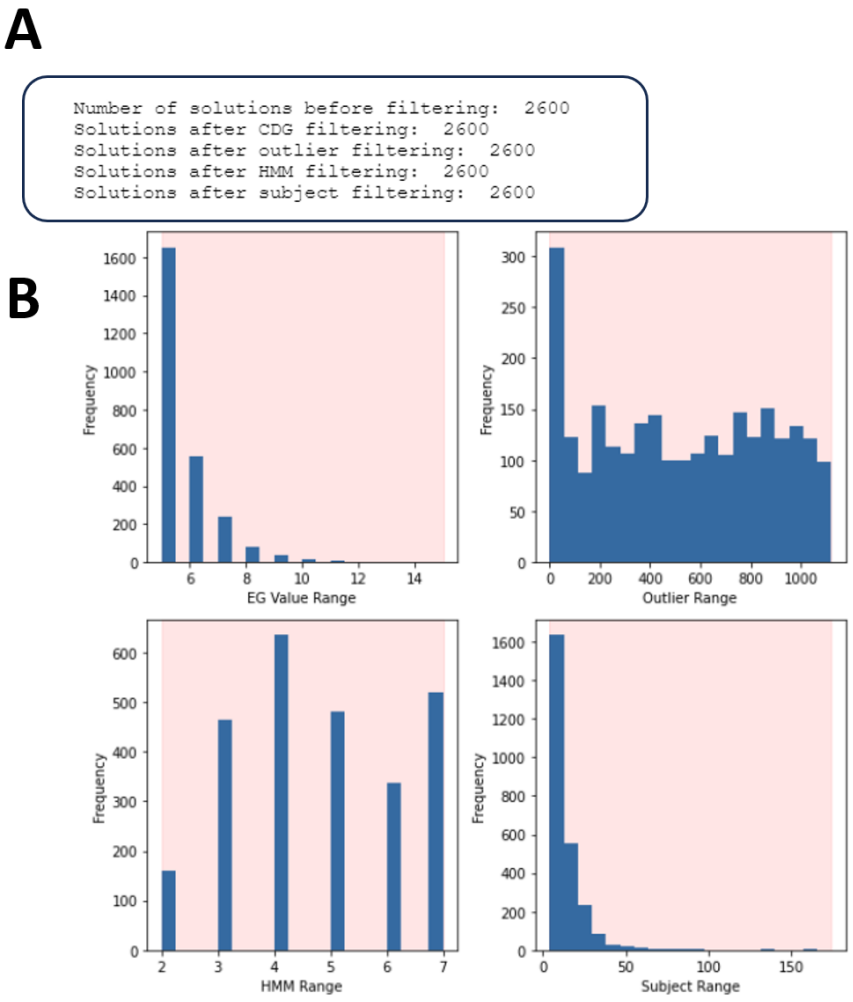

Figure 4.17: Output of the GUI after parameter selection.

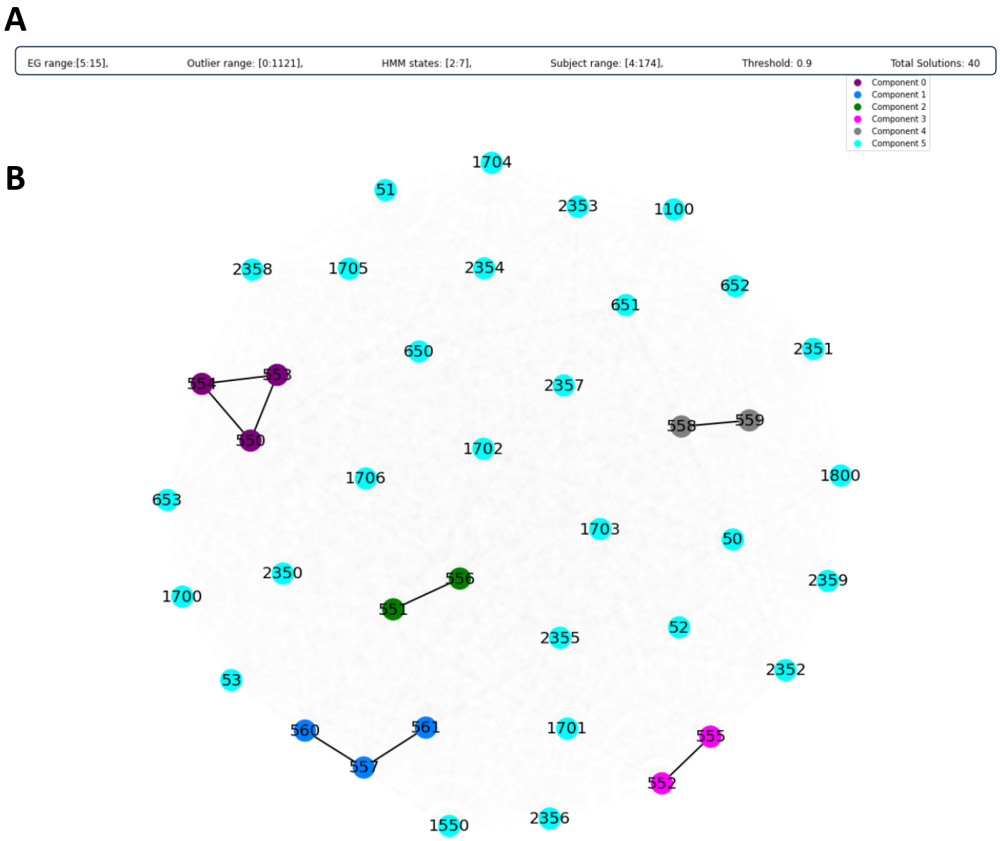

Figure 4.18: Output of the GUI after parameter selection.

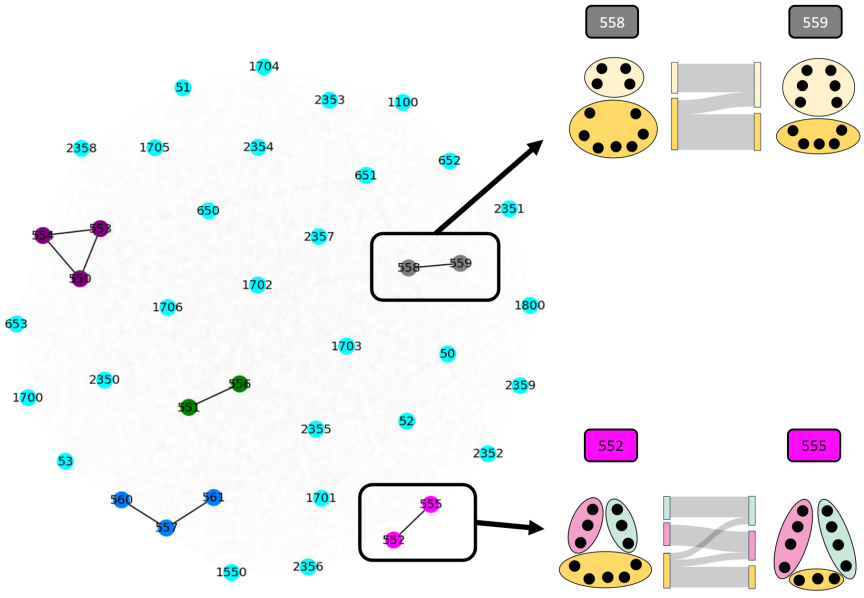

Figure 4.19: Output of the GUI after parameter selection.

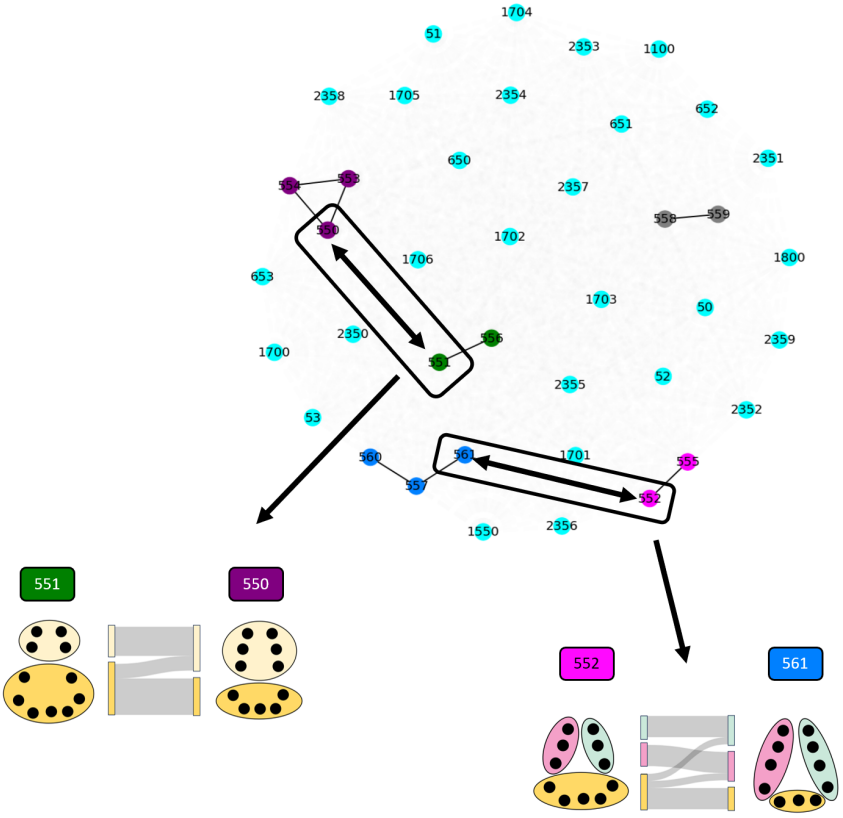

Figure 4.20: Output of the GUI after parameter selection.

4.5 Procedure for Selecting and Analysing Clustering Solutions Using the GUI

Analysing clustering solutions using the GUI involves several steps that are implemented in the next chapter. The first step is to select a subset of the solution dataset. For our experiments, we select the top 5^{th} percentile of solutions based on the internal clustering validation measures from the dataset of all available solutions. Next, using the GUI, one can specify the desired range of parameters for the clustering solutions. This filtering process reduces the dataset, allowing the selection of a specific subset of high-quality clustering solutions. After filtering, the graph is plotted for the selected solutions using a similarity threshold, which can be adjusted based on the required level of similarity for the task.

The GUI is designed for an *analyst* exploring many heterogeneous collections of clustering solutions to identify equivalence and redundancy. In this descriptive setting, a single automatic threshold does not generalise due to the edge-weight distribution in the solution graph, depending on dataset size, the diversity of solutions in $\mathcal{F}$, and the scale of the similarity measure. Allowing the analyst to choose θ "by eye" leverages domain context (e.g., favouring conservative equivalence classes vs. broader groupings) and adapts to known families of solutions that should or should not be merged. Manual selection introduces subjectivity and may reduce reproducibility. Analysts using the tool should record each run as a reproducible snapshot, saving the the chosen threshold θ and any preprocessing choices, together with a brief rationale for θ and a stability range $[\theta_{\min}, \theta_{\max}]$ over which components persist.

Higher threshold values mean that clustering solutions need to be extremely similar to belong to the same connected component. Lower threshold values produce fewer connected components, grouping solutions with greater variation together. The choice of threshold, however, depends on the dataset's distribution of similarities. If most solutions are dissimilar, a higher threshold may be needed to reveal structure in the graph. On the other hand, in datasets with more similar solutions, a lower threshold may be enough.

It is recommended to experiment with different threshold values until the desired level of connectivity is achieved. Once the appropriate threshold is identified, the results can be visualised. This includes both the original clustering solutions and the Sankey diagrams for comparing clustering solutions within the same connected component and across different components. This step is crucial for understanding the relationships between different groups of solutions, as well as the changes that occur between specific clusters in different solutions.

The next step in the process is to select prototype solutions from each identified connected component. The prototype solution is the clustering result with the highest internal validation index within its component. Certain solutions with lower internal validation indices may remain unconnected. The solutions from each component are plotted against their respective validation indices, allowing for the inspection of disparities between different components.

4.6 Example Case of GUI

To understand the SNACk dataset clustering solutions, we used the Graph GUI. In this graph, nodes are colour-coded to represent different components: Component 0 in grey and Component 1 in cyan. The edges, representing similarity, are thicker and more solid for highly similar solutions. Notably, nodes 50, 51, 52, and 55 form a tightly connected subgraph in Component 0, indicating these solutions are highly similar and suggesting a stable clustering region with robust patterns. In contrast, nodes 550 and 2350 are isolated within Component 1, signifying these solutions are distinct from each other and from those in Component 0. The visualisation includes key parameters such as the threshold for clustering similarity set at 0.7, and ranges for EG, outliers, HMM states, and subjects, providing context for the solutions being analysed. The total number of solutions visualised is 5. This graphical representation effectively highlights both groups of similar solutions and distinct outlier solutions, facilitating the assessment of clustering stability and diversity.

All solutions in the graph of 5 solutions are presented in the graph below Figure 4.22, atop the nodes in the graph, we add the Sankey diagram of all sequences for the corresponding solution.

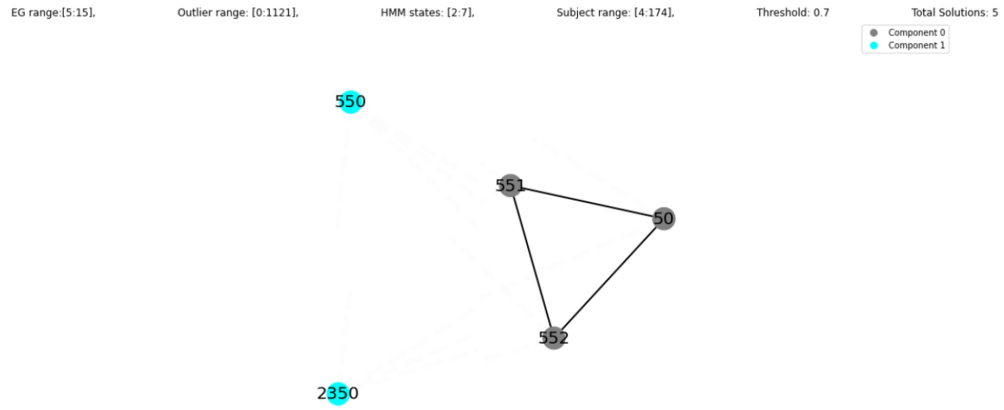

Figure 4.21: Solution graph from the SNACk dataset using 5 solutions and a threshold of 0.7.

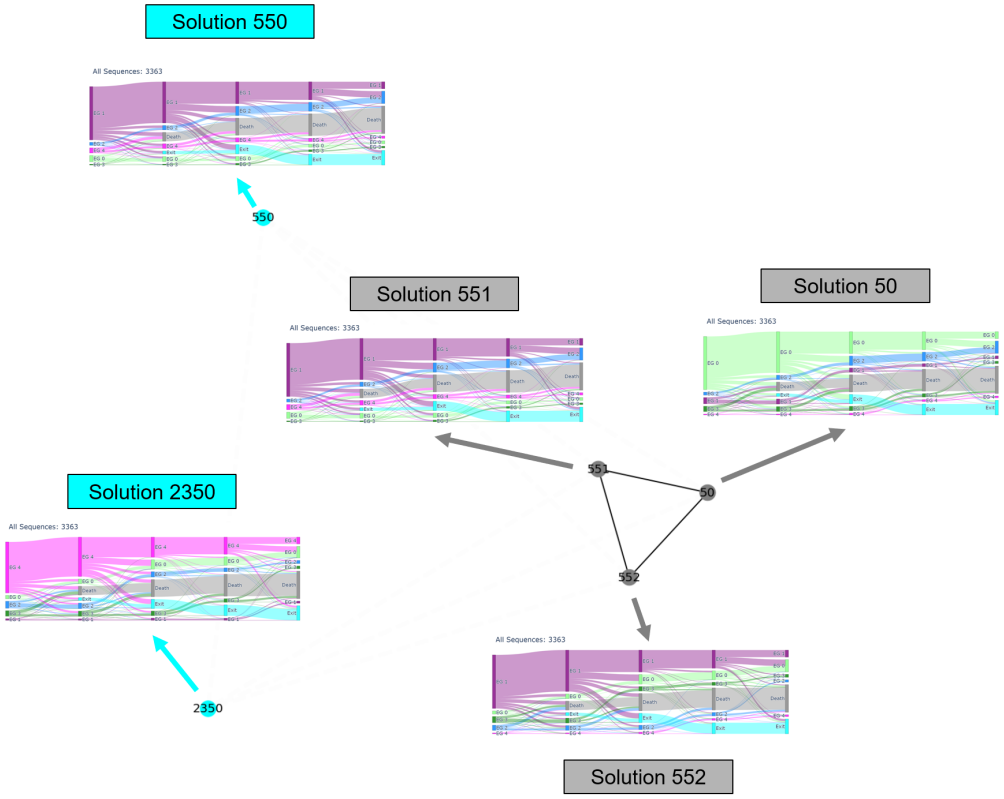

Figure 4.22: Graph with all solutions.

1. Sankey of all 5 solutions Figure 4.23
2. Connected Component
   (a) Subject Clusters Sankey comparisons Figure 4.24
   (b) Event Group Sankey comparisons Figure 4.25
3. Unconnected Solutions
   (a) Subject Clusters Sankey comparisons Figure 4.26
   (b) Event Group Sankey comparisons Figure 4.27

The sankey diagrams represent the relationship across different clustering solutions, similarly to how they were presented in Figure 4.7. In the figure below, we show how the 5 solutions from this graph differ in terms of their subject clusters. Along with this, we also present the same type of Sankey diagram for the Event groups across different clustering solutions.

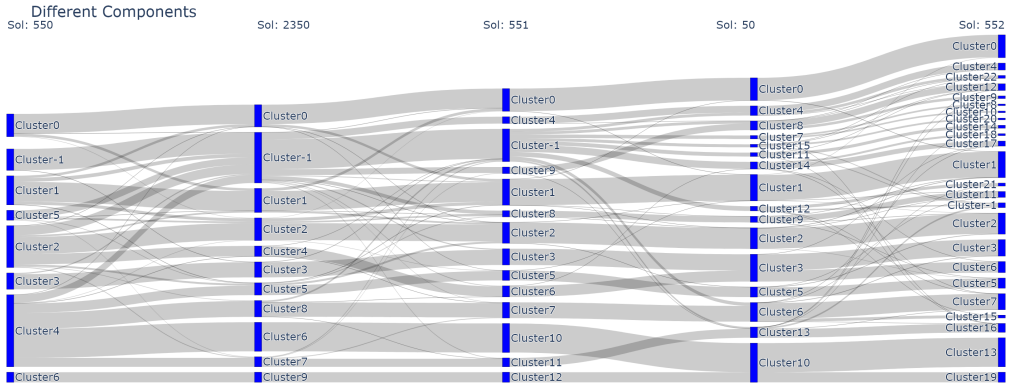

Figure 4.23: Sankey for subject clusters in all 5 solutions.

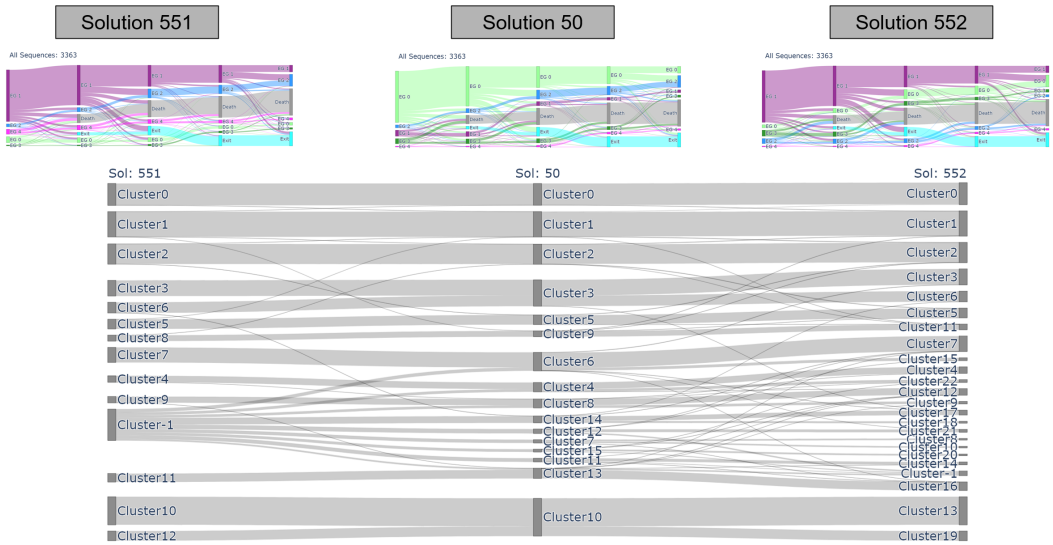

Figure 4.24: Sankey for subject clusters in the connected component.

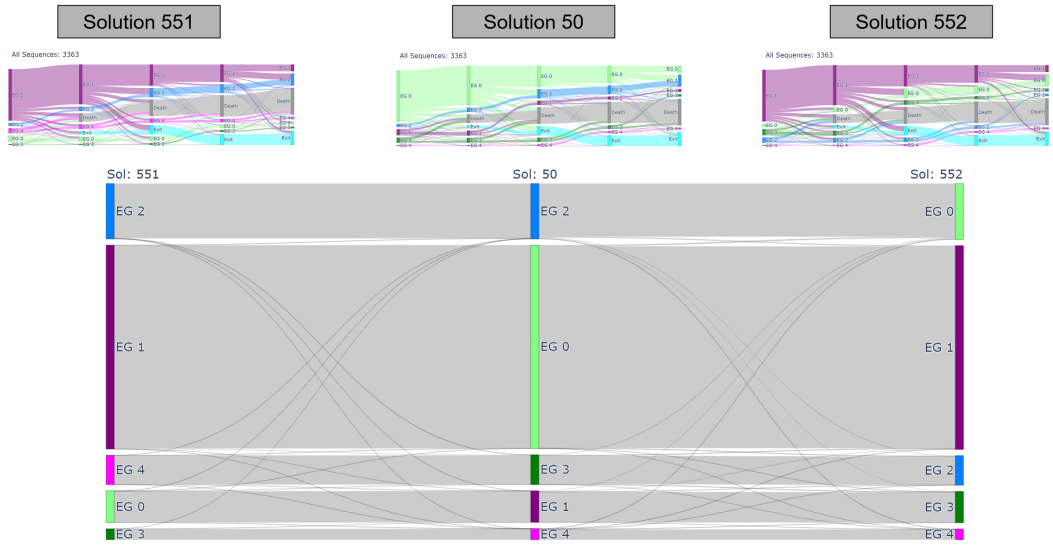

Figure 4.25: Similarly to the Subject Cluster Sankey diagrams, we also compare the Event Groups between different solutions in the connected components.

4.6.1 Unconnected Solutions

Unconnected solutions refer to clustering solutions that do not belong to any connected component in the graph. These solutions are significantly different from others, indicating unique groupings or patterns that are not shared with other solutions. Analysing unconnected solutions can provide insights into outlier behaviours or alternative structures within the data. The Sankey diagram for subject clusters within the unconnected component is presented in Figure 4.26.

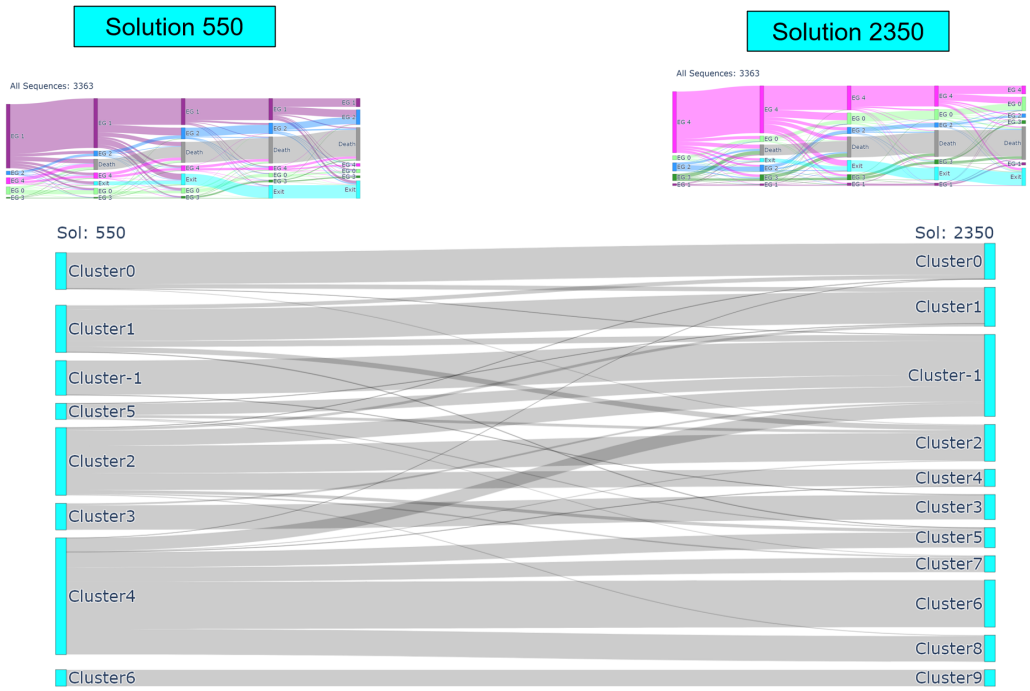

Figure 4.26: Sankey for subject clusters in the unconnected solutions.

4.6.2 Event Group Sankeys for Different Solutions

We can also present the Event Groups as a Sankey diagram between different solutions. In Figure 4.27, we can see that the Event groups EG 1 and EG 4 are largely the same in the different solutions.

The graphical representation, enhanced by Sankey diagrams that illustrate transitions across different clusters, enabled an intuitive exploration of the solution space, highlighting both the consistency and diversity present within the clustering outcomes. This approach is instrumental in guiding informed decisions on which clustering solutions to investigate further, ultimately contributing to a deeper understanding of patient health trajectories and multimorbidity patterns.

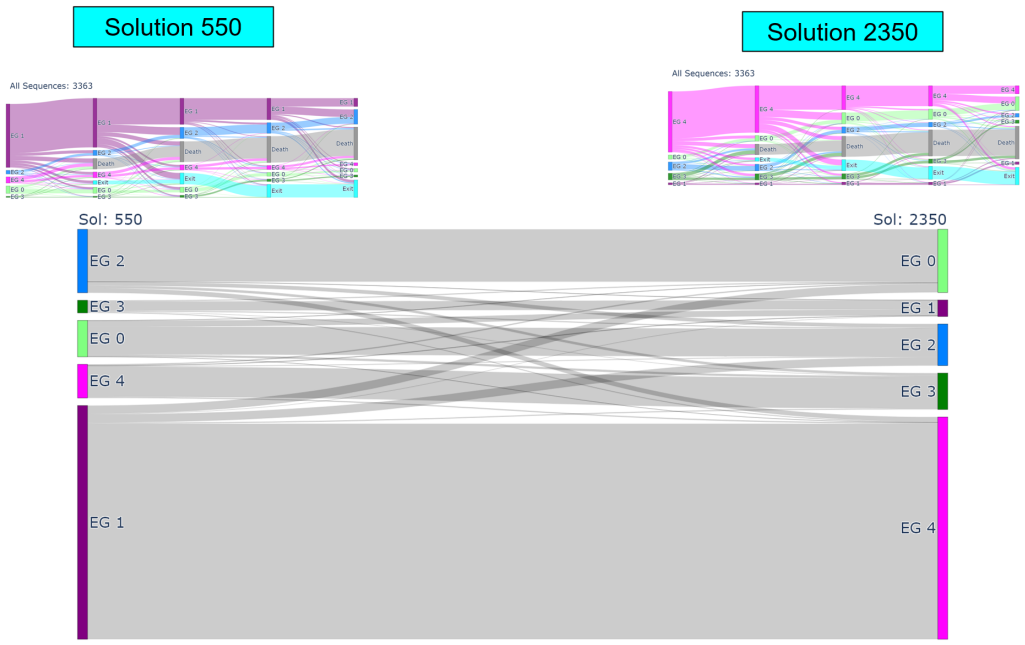

Figure 4.27: Sankey for Event Groups in the unconnected solutions.

Chapter 5

Application to Multimorbidity Data

Having developed a model for clustering multivariate sequence data as well as analysing and extracting distinct, unique solutions, we apply this methodology to two sets of real-world medical datasets of patient observations. Both datasets record the conditions patients were diagnosed with throughout the observation period. Using these datasets, we show that our model is able to capture relevant sets of timepoint clusters of similar observations of conditions. These clusters of timepoints within the context of the medical datasets provide insight into the types of conditions occurring to different subjects throughout their respective observation periods. This allows us to then describe real-world subjects using our derived timepoint clusters and analyse the progression of their health using commonly occurring sets of conditions within the timepoint clusters. The sequence clustering stage groups subjects into subgroups of disease progression. We aim to show that these subgroups reflect their common underlying state, which was previously unintelligible if looking directly at the raw observation data. We believe that these subgroups of subjects and their common health trajectories can, in the future, be used to investigate hidden insights into disease progression, treatment efficacy, and patient outcomes, informing evidence-based decision-making in clinical practice.

We applied our 2-Stage Clustering pipeline to the SNAC-k and CARE75+ datasets. The hyperparameter optimisation for the pipeline was run with a preselected range of hyperparameters and initialisation parameters. After selecting high-quality solutions from the hyperparameter optimisation results, we applied our ClusterView graphical user interface (GUI) methodology to better understand the variety of clustering solutions. Using the tools described in Chapter 4, we select several prototype solutions from the results of the 2 datasets and analyse these in detail in the remainder of the chapter. All supplementary figures and result images are provided in the online appendix GitHub (Thesis_appendix).

5.1 Datasets

The two datasets used in our case studies were both longitudinal cohort studies, tracking repeated measures of condition diagnoses over multiple timepoints. This structure allows for the examination of multimorbidity progression in elderly populations. The datasets provide valuable insights into trends and patterns over time, particularly useful for understanding

long-term health changes. Both datasets focused on elderly individuals with multimorbidity, enabling us to investigate whether these individuals could be grouped into distinct subpopulations based on their health trajectories. The number of subjects, timepoints, and unique conditions recorded in these datasets is shown in Table 5.1.

Dataset	Number of Subjects	Total Timepoints	Diagnoses Recorded
SNAC-K	3363	5	60
Care 75+	1278	6	35

Table 5.1: Description of 2 real-world datasets used in the case studies of the 2-stage clustering pipeline.

5.1.1 SNAC-K dataset

The Swedish National Study on Ageing and Care in Kungsholmen (SNAC-K) is a longitudinal research project focused on understanding the ageing process and the various factors that influence health and well-being in older adults `https://www.snac-k.se/`. Conducted in the Kungsholmen district of Stockholm, Sweden, SNAC-K is part of a larger national initiative, the Swedish National Study on Ageing and Care (SNAC), which includes several regions across Sweden. The primary objectives of SNAC-K are to examine the ageing process by investigating the physiological, psychological, and social changes that occur as people age; identify risk and protective factors to understand what contributes to healthy ageing and the development of age-related diseases; promote public health by providing insights that can inform strategies and policies aimed at improving the quality of life for older adults; and advance scientific knowledge by contributing to the global body of knowledge on ageing through high-quality research and data sharing. The study involves collaboration among researchers from various disciplines, including gerontology, neurology, psychiatry, epidemiology, and social sciences. This multidisciplinary approach ensures a holistic understanding of the ageing process.

5.1.1.1 Study Design

SNAC-K employs a longitudinal cohort design, which involves repeated observations of the same individuals over an extended period. This design allows researchers to track changes over time and identify trends and patterns related to ageing. Participants are randomly selected from the population of older adults living in Kungsholmen. The study includes several cohorts, each representing different age groups, ranging from those in their 60s to those over 100. Participants who were under the age of 78 were assessed every 6 years, while those who were aged 78 or older were assessed every 3 years. During each follow up the participants undergo a 5 hour-long clinical and functional assessment. The information collected includes: diagnoses via physical examination, medical history, examination of medical charts, self-reported information, drug information, and inpatient and outpatient care data. Some subjects died before completing the study, while others quit the study early for personal reasons. All diagnosed conditions were categorised into 60 chronic conditions categories [138].

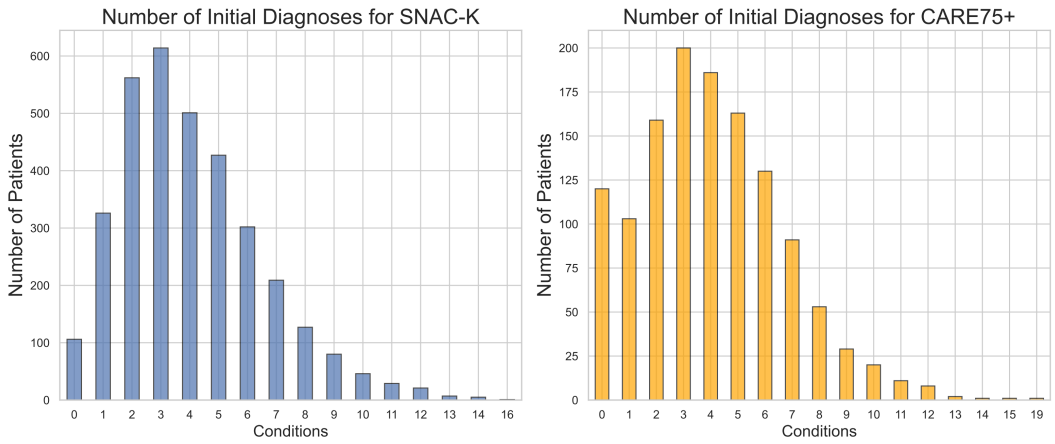

Figure 5.1: SNAC-K Total initial diagnoses.

In addition to information about what conditions the subjects were diagnosed with, we were also given access to age and gender information. The age-pyramid of the SNAC-K dataset is shown in Figure 5.2.

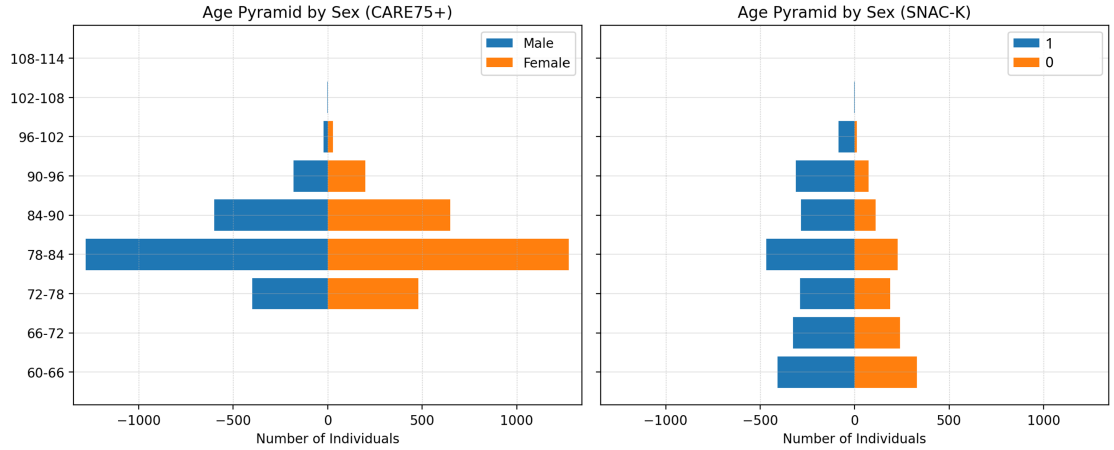

Figure 5.2: SNAC-k and CARE75+ age pyramids.

5.1.2 The Community Ageing Research Study (CARE75+)

The Community Ageing Research Study (CARE75+) is a longitudinal study designed to enhance the management of frailty through an integrated approach that spans primary care, secondary care, and social services. Frailty is characterised by a heightened vulnerability to poor health outcomes following stressor events. Effective management of frailty necessitates integrated care pathways supported by evidence-based interventions. However, the recruitment of frail older adults into clinical trials has been challenging, leading to a lack of robust evidence from clinical trials. CARE75+ addresses this gap by providing a compre-

hensive recruitment platform and observational data on older adults, aiming to recruit 2,500 participants.

5.1.2.1 Study Design

Participants are recruited via their GP practices and undergo cognitive, physical, and psychosocial assessments at baseline, 6, 12, 24, and 48 months. These assessments are conducted in participants' homes or optionally via telephone or the internet. Detailed demographic data, along with measures of frailty, quality of life, cognition, mood, daily living activities, resilience, loneliness, and comorbidities, are collected. Blood samples are taken at baseline and 12 months to support basic science research at the CARE75+ bio-bank. The study includes community-dwelling older adults aged 75 and above, while exclusion criteria encompass care home residents, bedbound individuals, those with terminal cancer, individuals with an estimated life expectancy of three months or less, and those receiving palliative care services [139].

For the purposes of our research, we extracted the sequences of conditions from the CARE75+ dataset. The sequences included 6 total time-points.The figures in Appendix C detail the total conditions diagnosed for each patient as well as how many instances of each specific condition were diagnosed at each timepoint across the entire dataset. Additionally, an age pyramid of the subject population in the CARE75+ dataset is presented in Figure 5.2.

5.1.3 Data preprocessing

Each dataset contains N subjects observed on a common discrete time grid of length T. At each timepoint, the subject's state is encoded as a binary row vector of length D (one indicator per condition). Observations are aligned to this grid and aggregated within each interval to produce presence/absence per condition. Missing timepoints are handled by *forward filling* per subject and condition: if $x_{d,t}^{(n)}$ is missing, set $x_{d,t}^{(n)} \leftarrow x_{d,t-1}^{(n)}$.

5.2 Application of 2-Stage Clustering pipeline to real-world datasets

The clustering of medical datasets is a critical task in healthcare analytics, enabling the identification of patterns and subgroups within patient populations. Effective clustering algorithms play a pivotal role in uncovering meaningful insights from large-scale medical data, contributing to improved diagnosis, treatment, and patient care. In this section, we present the results of hyperparameter optimisation for the clustering of two real-world medical datasets: SNAC-K and CARE75+. We apply our 2-stage pipeline to the 2 real-world datasets and present the results of the hyperparameter optimisation. By using hyperparameter optimisation, we aim to identify clustering solutions with meaningful subgroups that represent distinct phenotypes or clinical profiles within each dataset. To find the optimal solutions, we use the methodology described in Chapter 3, where we apply our 2-stage clustering pipeline using multiple initialisation parameters as described in Section 3.5.

In this section, we describe the datasets used for the analysis and present the results of applying the pipeline to both datasets. We highlight the solutions from the hyperparameter optimisation that achieved the best loss values during training. In the following sections, we present the timepoint clusters and the Sankey diagrams of subject clusters for both datasets. We consistently found common timepoint clusters across the two datasets, suggesting that the first stage of the pipeline identifies similarly organised conditions in both populations.

5.2.1 Dataset Imbalance

We were also interested in how the number of unique observations in a dataset would impact the results across different parameters of the pipeline. The frequency of events was highly imbalanced in the two datasets. Some conditions, such as hypertension and dyslipidemia, were observed in almost every individual at all timepoints, whereas other conditions, like chromosomal abnormalities, were rarely observed. Details of the number of occurrences of each condition in both datasets are presented in Appendix 6.1. We introduced an initialisation parameter, λ, which represents how many least occurring unique events are to be removed from the dataset. Conditions were ordered based on the frequency of their occurrence during the first observation period. We then set λ as the number of the least frequent events observed at the start of the observation period. For example, in the SNAC-K dataset, which includes 60 unique observations, a single hyperparameter optimisation run with initialisation parameter $\lambda = 5$ would involve performing the optimisation on the dataset after removing the five least frequent events. Removing the least frequent events helps in simplifying the dataset by reducing noise, allowing us to focus on more commonly occurring conditions that are likely to have a larger impact on clustering results. By varying λ, we can analyse how the presence or absence of rare conditions affects the overall clustering and identify patterns that might be overshadowed by less significant data points. Varying the λ can also allow for clustering analysis at different levels of granularity.

5.2.2 Results of Hyperparameter Optimisation

After preprocessing the data to match the appropriate format for the pipeline (as described in Section 3.4), we ran the pipeline on the **Sheffield SHARC HPC** using the **hyperopt**

python module. The goal of the hyperparameter optimisation is to enhance the clustering quality by identifying the parameter values that produce meaningful and distinct patient and timepoint clusters.

To achieve this, we optimised several parameters of the pipeline: the dimensionality of the data (number of MCA dimensions), the number of timepoint clusters (number of centroids in K-means), the total number of states in the HMM model, and the two DBSCAN parameters (ϵ and $min\_samples$). This process is detailed in Section 3.3.

The search space for the hyperparameters is summarised in Table 5.2, showing the parameter distributions used for the hyperparameter optimisation. The `Uniform` distribution ensures that all values within the specified range have equal probability, while the `Int Uniform` distribution ensures equal probability for integer values.

Table 5.2: The Hyperopt search space we used for hyperparameter optimisation. The distributions are noted in pseudocode corresponding to the distribution in the Hyperopt package.

Hyperparameter	Search Space
MCA (m)	`Int Uniform` [1, 10]
K-means centroids (k)	`Int Uniform` [2, 20]
HMM States (Q)	`Int Uniform` [3, 15]
`DBSCAN` (ϵ)	`Uniform` [0.1, 1.0]
`DBSCAN` (min samples)	`Int Uniform` [5, 20]

In addition to optimising the hyperparameters, we also explored different initialisation parameters, such as α and λ, as detailed in Section 3.5. Table 5.3 presents the values used for these initialisation parameters. Each hyperparameter optimisation was conducted using the same ranges for the hyperparameter distributions, but with all possible combinations of the initialisation parameters. The parameter λ controls the number of the least frequent events that are removed from the dataset, which helps reduce noise and focus on more commonly occurring conditions, and the α parameter is used to control whether the timepoint cluster or sequence cluster quality is emphasised during optimisation.

Parameter	Values
α	0.3, 0.4, 0.5, 0.6, 0.7
λ	0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20

Table 5.3: Settings used for the different initialisations of the hyperparameter optimisation.

Early Stopping Condition

To further enhance the efficiency of the hyperparameter optimisation process, we implemented an early stopping condition. The optimisation stops if there is no improvement greater than 0.01 in the combined loss value (w) for 200 consecutive evaluations. This threshold ensures that only meaningful improvements are taken into account, which prevents the optimiser from continuing when further gains are negligible. The early stopping condition plays a

crucial role in balancing the trade-off between finding the optimal solution and conserving computational resources. Without early stopping, the optimisation process might continue indefinitely, attempting to make small improvements that do not significantly enhance the quality of the clustering. By terminating the search when no substantial progress is observed, we prevent overfitting, reduce computation time, and make the entire optimisation process more practical. The results of the loss optimisation for all sets of initialisation parameters are shown in Figure 5.3.

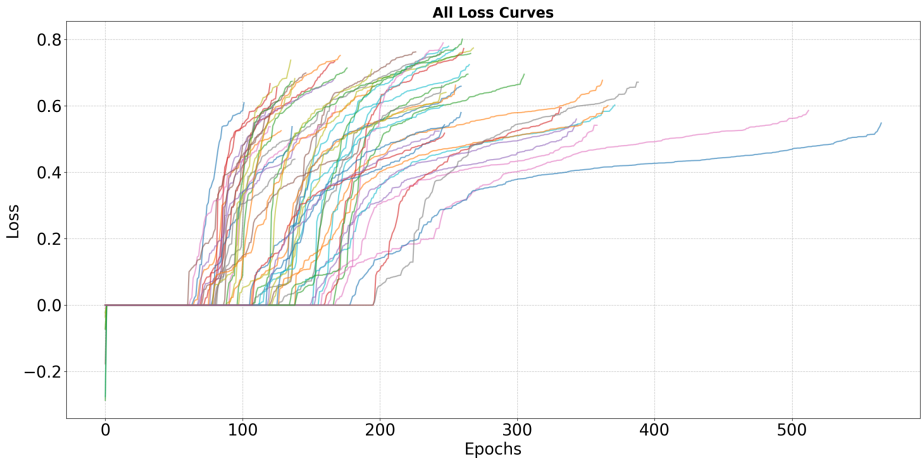

Figure 5.3: The loss results of the SNAC-k optimisation of each set of initialisation parameters.

After running the hyperparameter optimisation with all possible combinations of initialisation parameters, we selected the top 50 solutions with the highest w value from each initialisation run. These selected solutions were then combined and sorted based on the combined loss value w, which represents the overall quality of both the timepoint and sequence clusters. The loss value quantifies the performance of the clustering pipeline, incorporating both aspects of clustering quality. By focusing our analysis on the top solutions, we ensure that only the most effective clustering configurations, with the highest combined silhouette indices across all initialisation parameter sets, are considered.

5.2.3 Top SNAC-k results

After the optimisation of the SNAC-k dataset, and selection of the top 50 solutions with the highest w values, the final set of solutions had a total of 2600 solutions.

The top 5 solutions with the lowest loss values (w) across the entire dataset are presented. These 5 solutions, according to the w value, are the optimal clustering configurations. Table 5.4 summarises the hyperparameters for each of these 5 solutions, including the initialisation parameters λ and α. The top configurations in Table 5.4 vary in HMM states (Q), the number of k-means centroids (k), and MCA dimensions (m), yet achieve nearly identical loss. This indicates a relatively flat objective surface, where small changes in representational capacity (m, Q) or EG granularity (k) yield equivalent subject partitions. In the hyperparameter

search, k and m exhibited the most stable optima ($k = 5$ and $m = 5$): neighbouring settings produced highly similar partitions that met our quality criteria.

In this section, we focus on presenting Solution 2350, which had one of the best loss values, along with its associated event groups and Sankey diagrams. Solution 2350 was chosen for detailed presentation because it demonstrated a good balance between cluster separation (as indicated by the silhouette scores c_{sil} and s_{sil}) and meaningful clustering of patient trajectories.

Index	Loss	ID	EPS_SUBJ	HMM_S	KM	MCA	MIN_S_SUBJ	c_{sil}	s_{sil}	λ	α
550	-0.807381	351	0.000194	3.0	5.0	5.0	105.0	0.358016	0.999966	0	0.3
551	-0.807353	460	0.000998	4.0	5.0	5.0	86.0	0.358016	0.999925	0	0.3
50	-0.807107	22	0.000936	3.0	5.0	5.0	11.0	0.357040	0.999993	2	0.3
552	-0.799205	217	0.000838	4.0	5.0	6.0	24.0	0.330733	0.999979	0	0.3
2350	-0.798465	264	0.000354	4.0	5.0	5.0	123.0	0.335342	0.996946	20	0.3

Table 5.4: Table of 5 solutions with the best loss from the hyperparameter optimisation of the SNAC-k dataset. The table details the hyperparameters along with the initialisation parameters λ and α.

Solution 2350

Event Groups

The solution 2350 has 5 timepoint clusters as listed under the KM column in the table. Timepoint cluster 3 is shown in Figure 5.4. The format for the figure is the same as in Figure 3.6. In this example, the condition with the highest EGS score was Dementia; therefore, this timepoint cluster was named EG 3: Dementia.

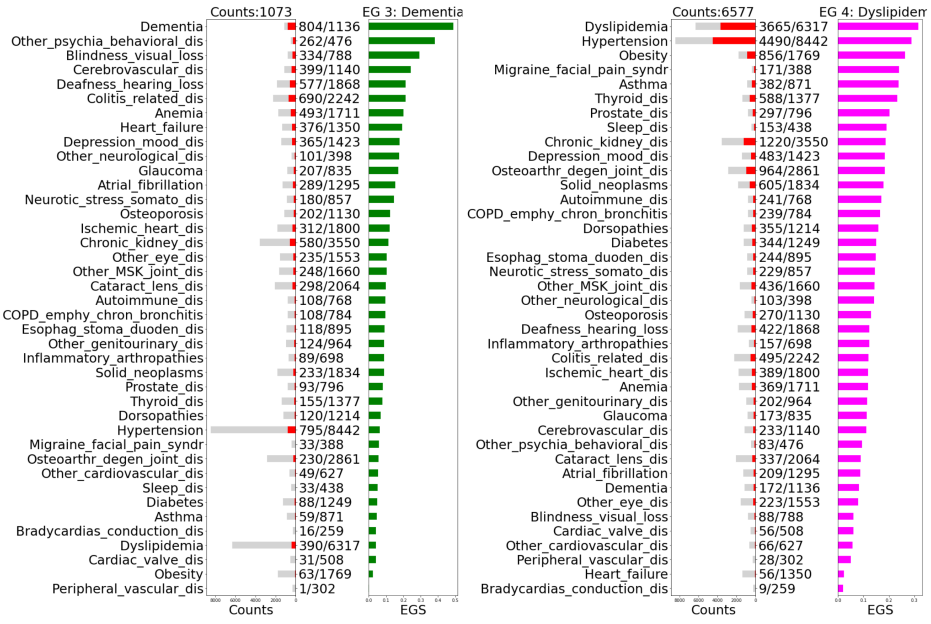

Figure 5.4: Barchart for EG 3 and 4 from solutions 2350.

The Sankey diagram for this solution is presented in Figure 5.5. This shows how all subjects in the dataset transitioned between the different EGs across the 5 observations. During the first observation of the dataset, more than half of the patients can be seen to be a part of EG: 4, and later split into different disease groups during later observations. Alongside the 5 event groups, the Sankey diagram shows that some patients either died or exited the study during the observation period.

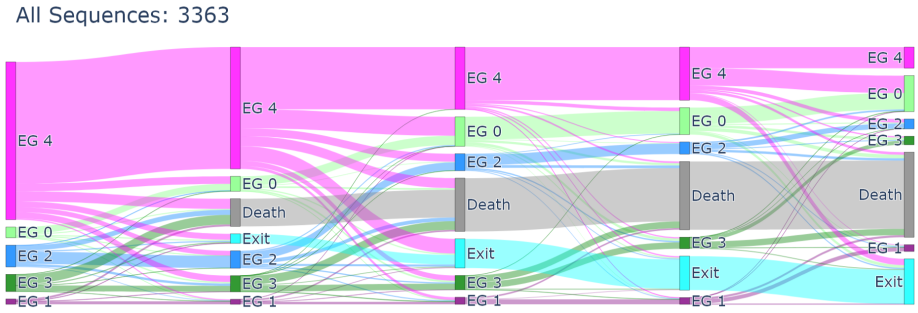

Figure 5.5: Sankey diagram showing all subjects in the SNAC-k dataset, reorganised using their EG assignments.

Figure 5.6: Sankey diagram showing sequence clusters 0 and 1. These clusters were mostly comprised of sequences that remained in the timepoint cluster 4. Sequence cluster 1 was different, as in the last timepoint, the sequences split into different timepoint clusters.

In Figure 5.6, we can see that all timepoints in sequence cluster 0 are Event Group 4 timepoints.

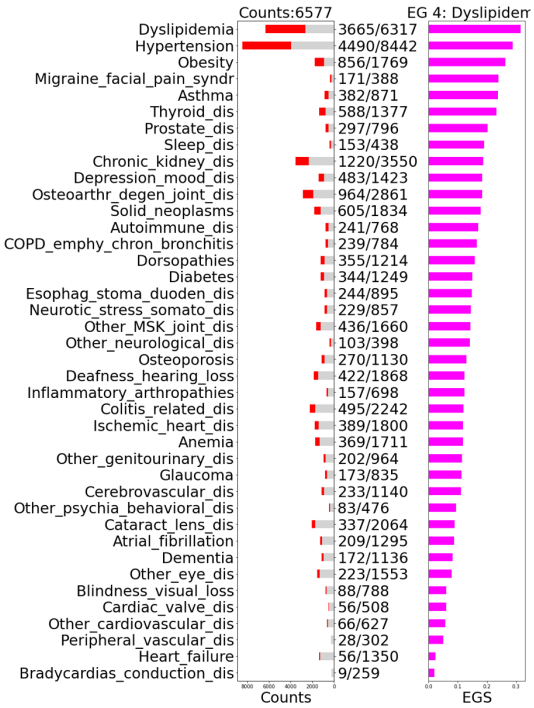

Figure 5.7: Barchart for Event Group 4 from solutions 2350.

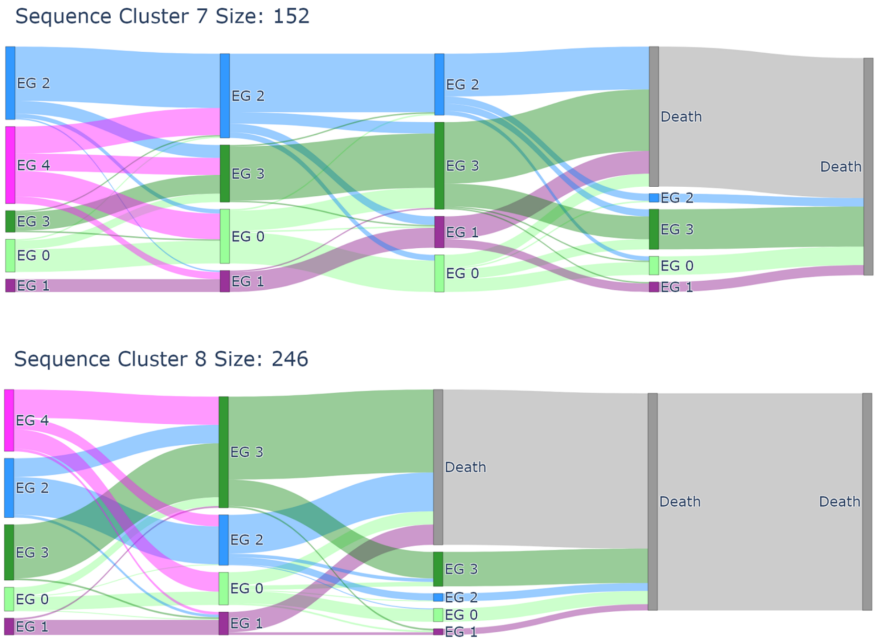

Figure 5.8: Sankey diagram showing sequence clusters 7 and 8. The sankey diagrams presented in this figure are much more varied compared to the sequence clusters shown in Figure 5.6. The sequences transitioned between many different timepoint clusters throughout the observation period.

The sequences were clustered into 9 clusters, several of which are shown in Figure 5.6 and Figure 5.8. The full details of this specific solution (2350), along with the other solutions from Table 5.4, are detailed in the Appendix.

SNAC-k Cohorts

Recruitment for the SNAC-k study occurred in multiple waves or cohorts, with each cohort being assessed five times. Since each cohort had a different initial age, we can categorise each cluster based on the respective cohort. An example of splitting the sankey diagram based on the age of the cohort is presented in Figure 5.9.

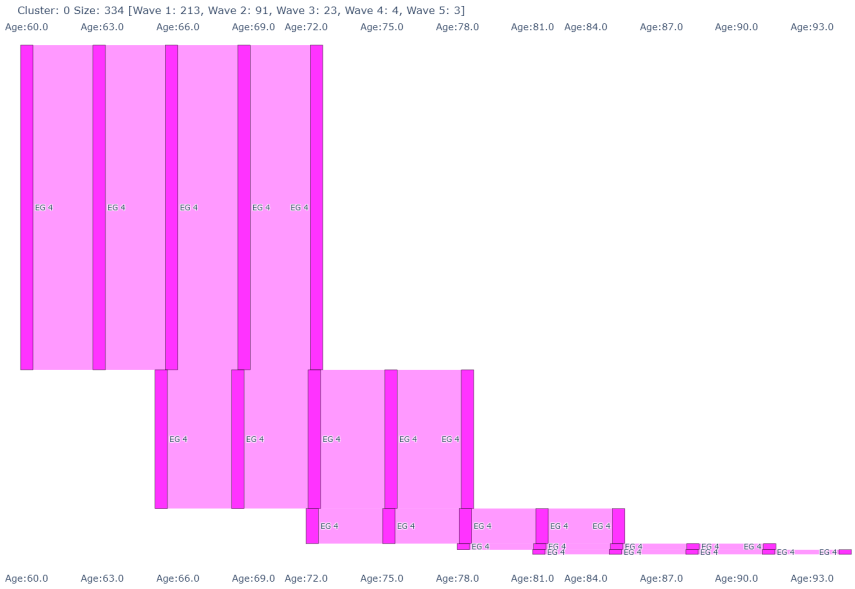

Figure 5.9: Cluster 0 of Solution 2350 from the SNAC-k dataset. The subjects from the cluster are separated based on their age at the beginning of the study.

The all-subjects Sankey (Figure 5.5) shows that over half of participants begin in EG 4 and subsequently disperse into other event groups—consistent with a broadly cardiometabolic entry state that branches as additional conditions accrue. The bar chart for EG 4 (Figure 5.4) confirms its composition is dominated by dyslipidemia and hypertension. Sankeys of individual clusters reveal trajectory heterogeneity: sequence cluster 0 remains entirely in EG 4 across all waves (persistence in a stable, low-complexity profile), whereas **sequence cluster 1** starts in EG 4 but splits at the final wave (a late divergence into a more complex state), see (Figure 5.8). More varied clusters (e.g., **clusters 7 and 8**) cycle through multiple EGs (Figure 5.8). Age-stratified sankeys (Figure 5.9) show that clusters with long residence in EG 4 skew younger at baseline, while earlier branching clusters include older cohorts.

5.2.4 CARE75+ results

In this section, we present the results from the hyperparameter optimisation of the CARE75+ dataset, focusing on the solutions with the highest combined loss values. These results reflect the most effective configurations for capturing meaningful patterns and subgroups within the dataset. Table 5.5 provides the details of the top 5 solutions, including their respective hyperparameters and initialisation parameters.

In Figure 5.10 we show the sankey diagram for all sequences from solution 1900. One of the largest event groups, EGs 4, is detailed in Figure 5.11.

Index	Loss	ID	EPS_SUBJ	HMM_S	KM	MCA	MIN_S_SUBJ	c_{sil}	s_{sil}	λ	α
1900	-0.788474	385	0.000176	6.0	5.0	5.0	17.0	0.294943	0.999987	14	0.3
1901	-0.788463	417	0.000461	7.0	5.0	5.0	29.0	0.294943	0.999972	14	0.3
1902	-0.788383	490	0.000026	5.0	5.0	5.0	15.0	0.294943	0.999857	14	0.3
1903	-0.787875	257	0.002150	6.0	5.0	5.0	50.0	0.294943	0.999132	14	0.3
1904	-0.787480	138	0.005277	2.0	8.0	5.0	147.0	0.295338	0.998397	14	0.3

Table 5.5: Table of the 5 solutions from the CARE75+ optimisation with the best final loss values. The table details the hyperparameters along with the initialisation parameters λ and α.

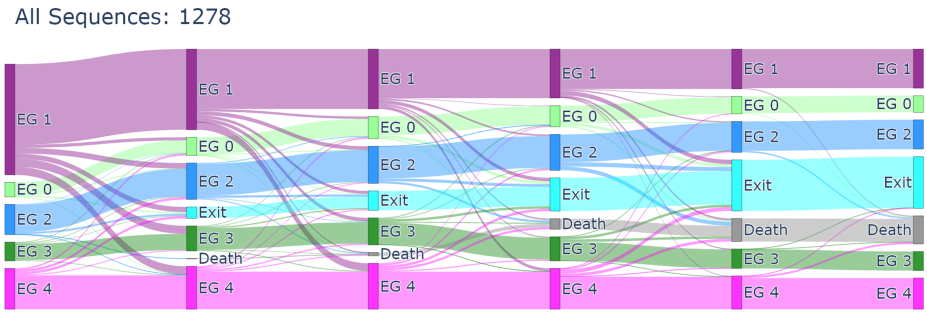

Figure 5.10: Sankey diagram showing all subjects in Solution 1900 of the CARE75+ dataset, reorganised using their event group assignments.

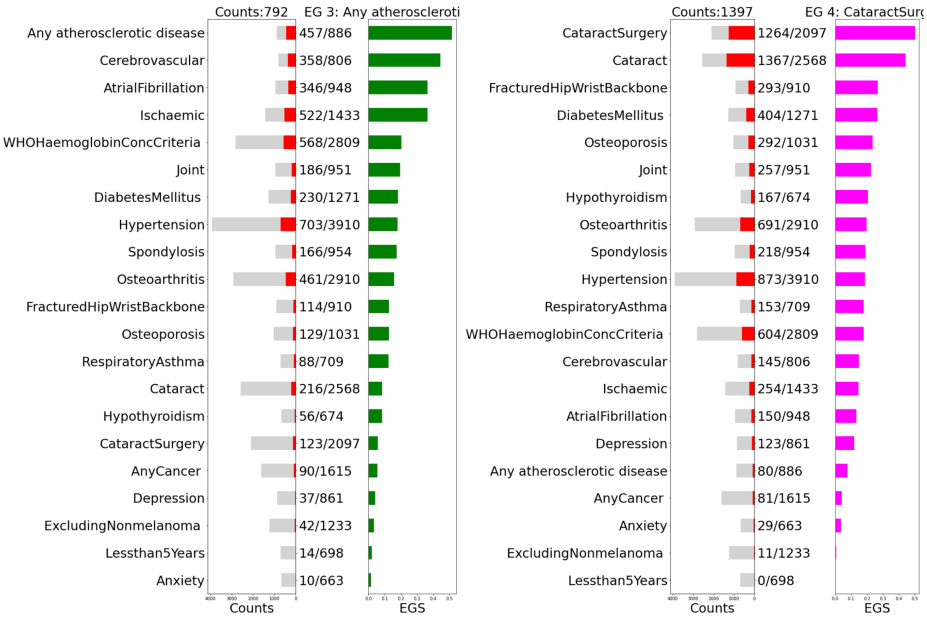

Figure 5.11: Barchart for Event Groups 3 and 4 from solution 2350.

5.2.5 Interpretation

Across both datasets, we observed recurrent timepoint clusters with closely matching condition profiles. In particular, each cohort contains an Event Group dominated by ocular disorders (SNAC–K: EG 4 with cataract/cataract surgery; CARE75+: EG 0 with cataract–lens disease, other eye disease, and glaucoma), indicating a shared multimorbidity structure despite differences in data source and parameterisation (see Figure 5.12). These cross-cohort recurrences suggest that a subset of EGs is reproducible across the two populations, providing a stable basis for subsequent trajectory comparison and interpretation. A recurrent cardiometabolic EG (dominated by dyslipidaemia and hypertension) represents a low-complexity profile commonly occupied early and often. An "ocular" EG (cataract, glaucoma, other eye disease) appears in both cohorts, consistent with age-related ophthalmic burden. Trajectories that persist within a single EG over multiple waves (e.g., cardiometabolic) are consistent with chronic disease; trajectories that diverge from a broadly cardiometabolic EG into more complex EGs (ocular, cardiac, cancer-related) are plausible with ageing. Common multimorbidity motifs in older adults include cardiometabolic combinations (hypertension with dyslipidaemia and/or diabetes) and sensory/musculoskeletal patterns (e.g., arthritis with cataract/visual impairment); our EGs align with these patterns [5, 51, 140, 141].

The all subject Sankey for **Solution 1900** (Figure 5.10) similarly exhibits a one-to-many pattern from an initial large EG into diverse trajectories, with more prominent ocular and cancer-related branches in later waves Figure 5.11.

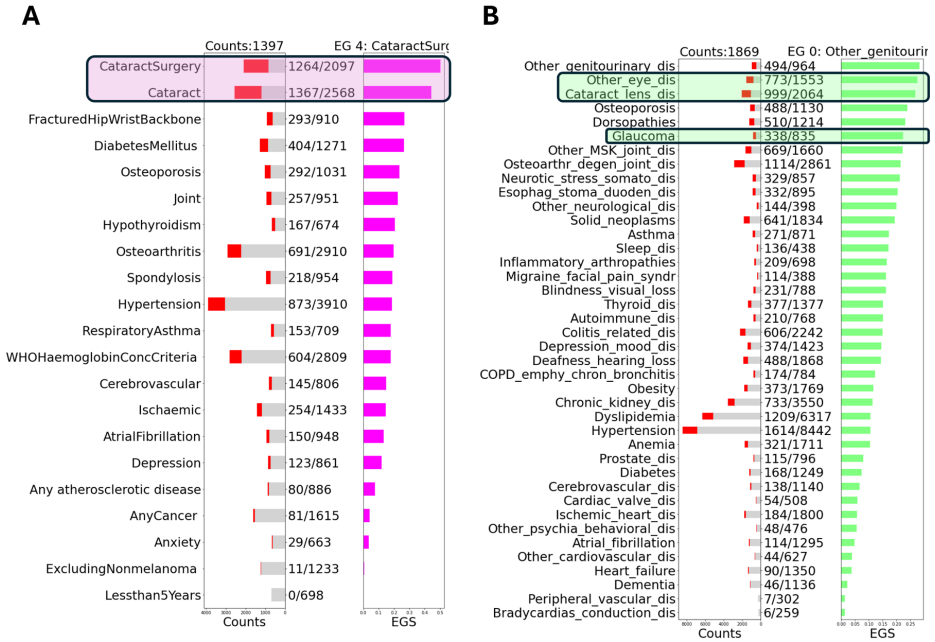

Figure 5.12: Event groups sharing similiar distributions of conditions, left (CARE75+) and right (SNAC-K).

5.3 Results of ClusterView Application

The analysis of clustering solutions for the SNAC-k and CARE75+ datasets follows a structured methodology using the ClusterView GUI, enabling us to effectively group and explore multiple equivalent clustering outcomes. During hyperparameter optimisation, it is common to encounter numerous equivalent solutions—distinct hyperparameter sets that yield nearly identical clustering results. This equivalence can lead to redundancy and challenges in identifying unique and informative solutions. The methodology introduced in the previous chapter aims to address these challenges through the use of ClusterView and prototype selection.

5.3.1 SNAC-k ClusterView Analysis

First, we selected the top 5th percentile of clustering solutions based on the internal validation measures. This selection ensured that only the highest-quality solutions were included in the analysis. Additionally, the number of individuals classified as outliers was restricted to a maximum of 30%, thereby excluding solutions with excessive outliers. The parameter selection process is illustrated in Figure 5.13.

Figure 5.13: Parameters for the selection of results

The similarity threshold θ was selected manually by inspecting the solution graph G_θ (retaining edges with $S_{ij} \geq \theta$). At each step of θ, the following were examined: (i) the number of connected components and the size of the largest component, (ii) the edge density to avoid trivial fully connected or overly fragmented regimes, and (iii) the interpretability of components. We chose the *smallest* θ for which components were both interpretable and *stable* under a small perturbation of θ (component memberships persisted when θ was nudged up/down). This was an ad-hoc decision (no automatic optimiser); discussion on the threshold choice is discussed in Chapter 4. Once the optimal threshold value was identified, we used it to construct and plot the similarity graph, as shown in Figure Figure 5.14. This graph visualises the connected components, each representing a group of similar clustering solutions.

After constructing the graph, each connected component was further analysed to identify the clustering solution with the highest internal validation index. These highest-scoring solutions, known as prototype solutions, were plotted for each connected component, as depicted in Figure Figure 5.15.

Finally, we compared the prototype solutions across different components using Sankey diagrams to gain insights into the relationships between clustering solutions. This comparison is presented in Figure Figure 5.16, allowing us to visualise how different solutions relate to one another within the SNAC-k dataset and understand the equivalence and distinctions between them.

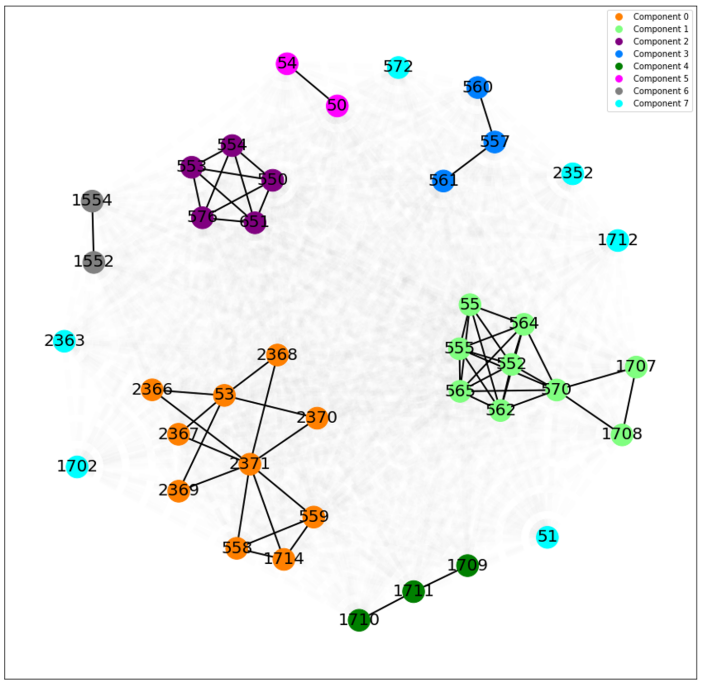

Figure 5.14: Graph for SNAC-k Threshold 0.75 40 Solutions

5.3.2 Discussion of Graph

The graph constructed using the SNAC-K dataset has 7 components of varying sizes. Component 3 has the fewest number of subject clusters, while component 6 has the highest number of clusters. This can be seen from Figure 5.16, where the Sankey diagrams of the highest loss values from each component are combined in a single solution Sankey diagram. The Sankey for solutions in Component 3, with the least number of clusters, is shown in Figure 5.17. The remainder of clustering solutions and Sankey diagrams between all components are in the Supplementary materials GitHub repository.

The highest number of subject clusters is Component 1 shown in Figure 5.18.

The Event groups across the different components are shown in Figure 5.19. The Event groups across all solutions from the graph appear to be largely similar, with slight variations across different solutions. The total number of Event Groups (EG) across all solutions is 5, while only 3 solutions had 6 EGs. The stability of the Event groups indicates that an optimal value of k was identified during optimisation to be between 5 and 6. The largest event group in each solution was the Event Group, comprised of occurrences of dyslipidemia and hypertension. This EG was primarily the first event in most sequences, indicating a general state of multimorbidity with few complications. When viewing the age distribution of subjects from different cohorts, the subject clusters which remained in the EG, consisting of dyslipidemia and hypertension, were significantly younger than subjects in clusters with more varied and complicated transitions between different EGs.

From all the selected solutions, we select solutions with the highest loss values as the **prototype solutions**. The diagram of this is shown in Figure 5.16. The selected solutions are analysed in detail in the following section.

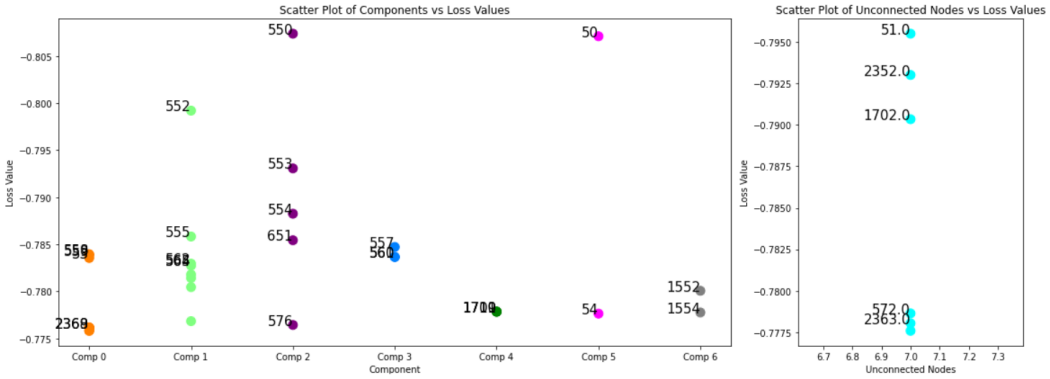

Figure 5.15: Loss of the different Components

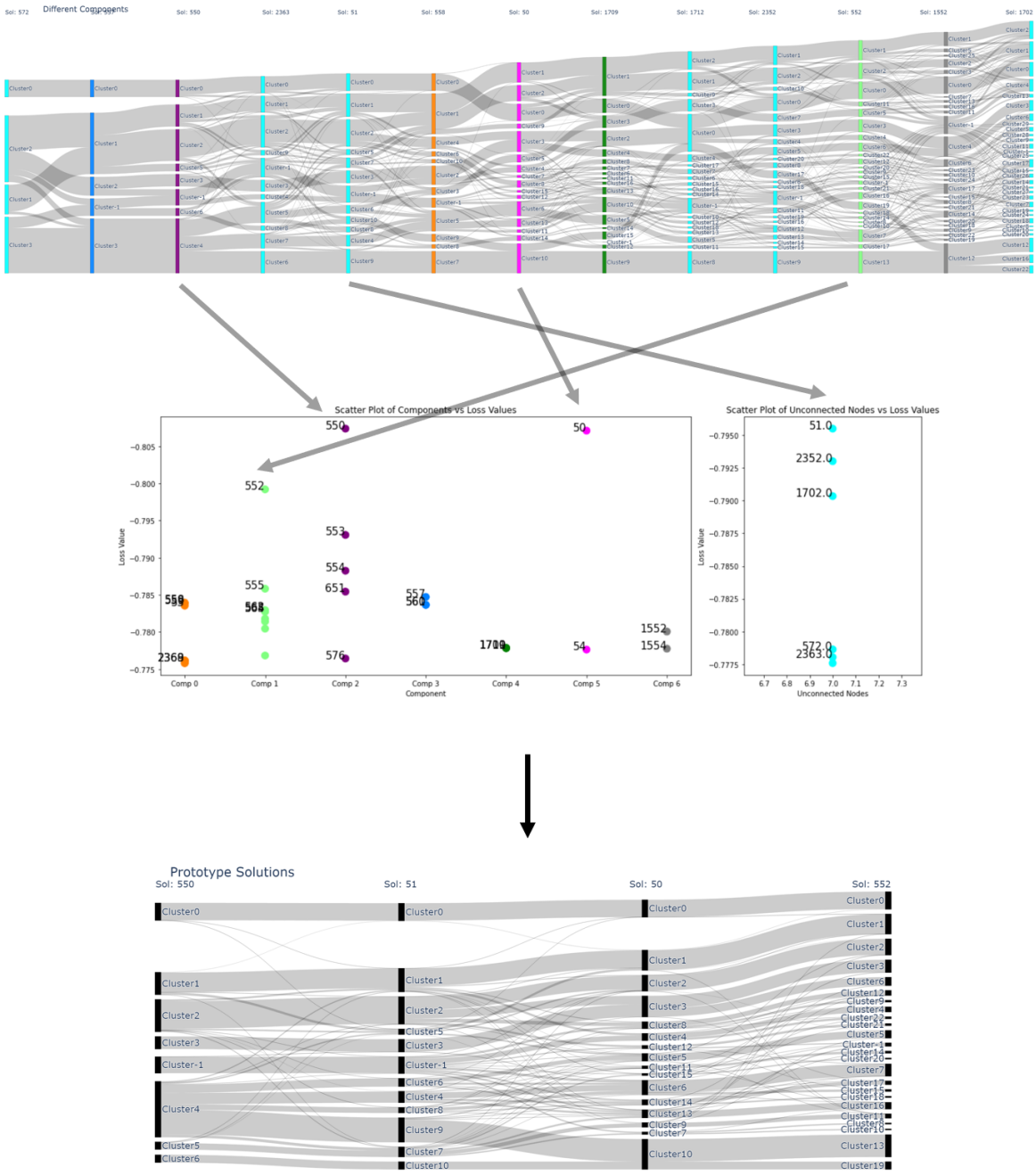

Figure 5.16: Loss of the different Components

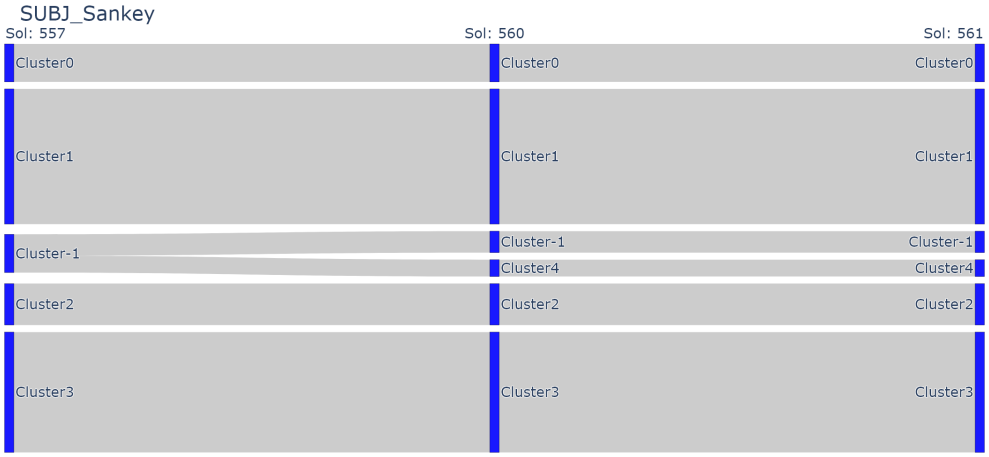

Figure 5.17: Sankey for Component 3 from SNAC-K graph.

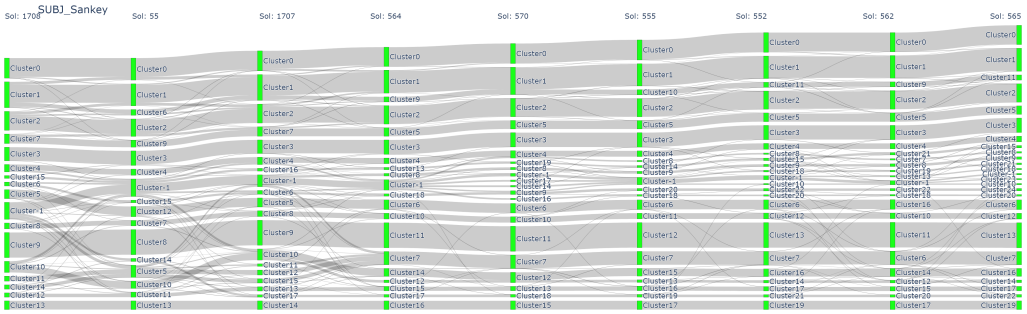

Figure 5.18: Sankey for Component 1 from SNAC-K graph.

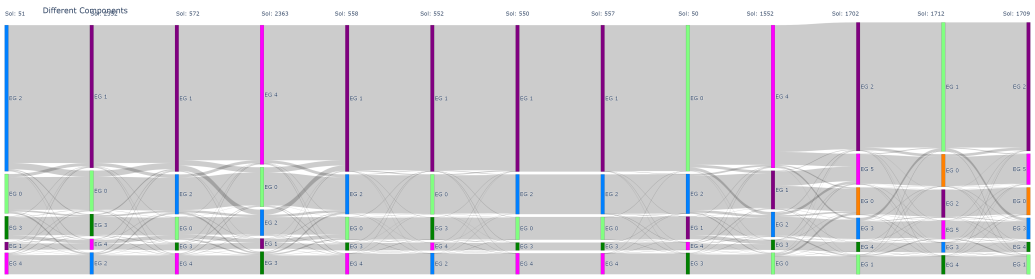

Figure 5.19: Sankeys for the EGs, across all prototype solutions and unconnected solutions (total of 13) from SNAC-K graph.

5.4 SNAC-K Prototypes

The 4 selected prototypes are largely distinct from one another as can be seen in the solution sankey diagram at the bottom of Figure 5.16.

Solution 552

Solution 552 has the highest number of subject clusters among all the prototype solutions. The solution contains five Event Groups shown in Figure 5.20 and Figure 5.21. As discussed previously, the largest event group is the one containing most of the observations of dyslipidemia and hypertension.

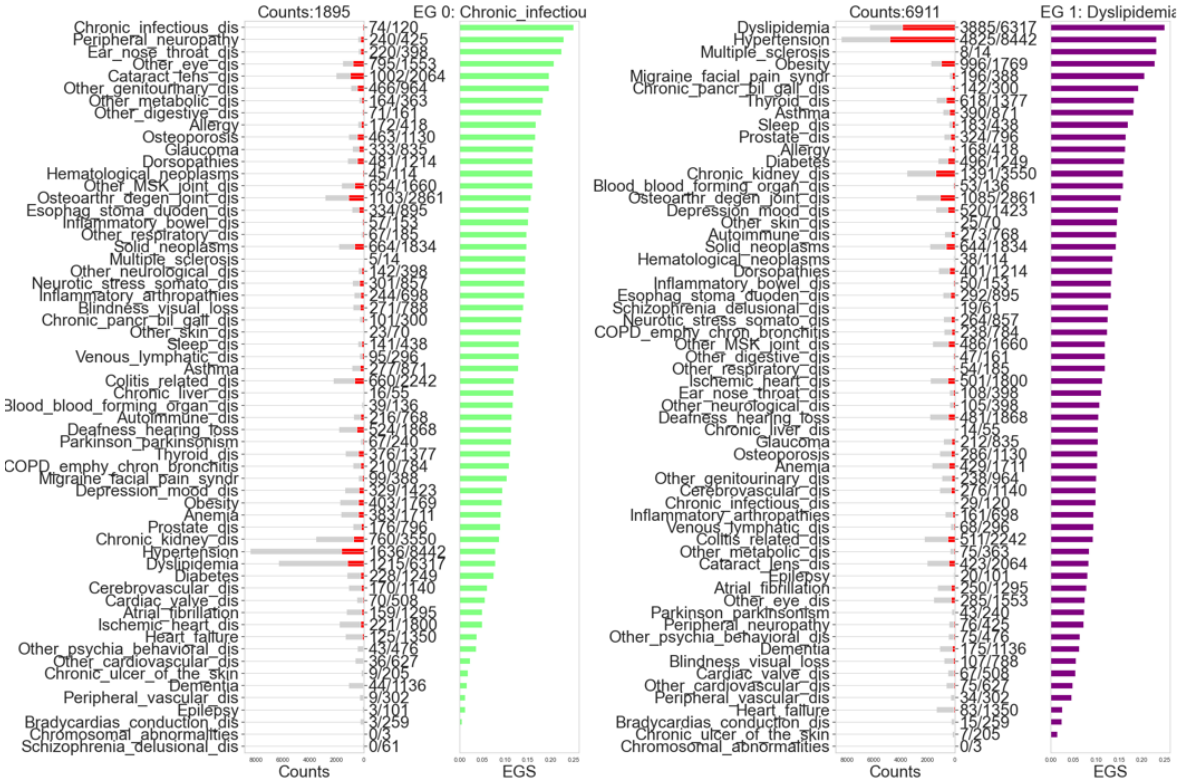

Figure 5.20: Event Groups 1 and 2 for Solution 552.

Solution 550

Solution 550 only contains 7 subject clusters and 5 event groups. Two clusters contain almost entirely transitions between the largest event group (EG 1) (containing most of the observations of dyslipidemia and hypertension) Figure 5.22.

Sequence clusters 2 and 3 are comprised of subjects that during the first 2 observations remained in the stable EG 1, but later either left the study, transitioned to more complex event groups or, in the case of Sequence Cluster 2, died.

For sequence cluster 4, the starting point is one of several possible EGs; however, the cluster appears to capture subjects that died early in the study, as seen in Figure 5.24. By

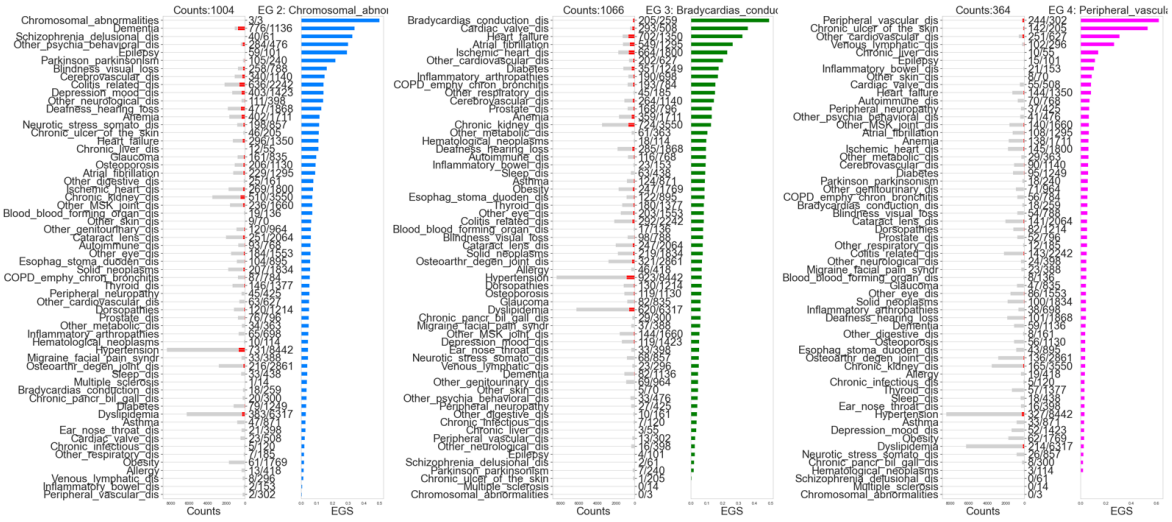

Figure 5.21: Event Groups 3, 4, 5 for Solution 552.

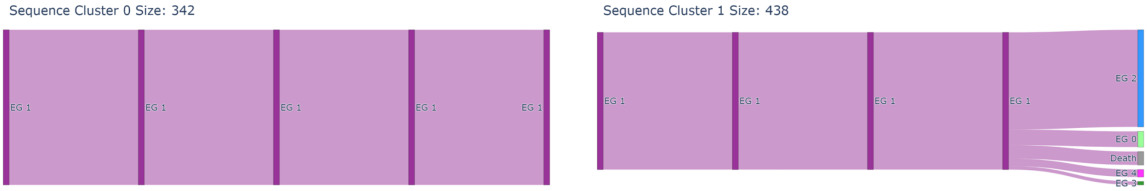

Figure 5.22: Sequence clusters 0 and 1 for Solution 550.

the end of the observation period, a large part of the cluster had died. As can be seen from the Sankey diagram separated based on different cohorts Figure 5.25, the cluster is mostly comprised of the oldest subjects in the study.

Solution 50

Solution 50 has a total of 5 event groups and 16 subject clusters. Subject clusters 0, 1 and 4 contain subjects that, for the majority observation period, remained in the EG 0 (containing most of the observations of dyslipidemia and hypertension). Other subject clusters also largely contained transitions between EG 0 at the early stages of the study and later splitting among the different EGs.

Solution 51

Solution 50 has a total of 5 event groups and 11 subject clusters. This solution contains a much higher variety of subject clusters. Only subject cluster 0 contains transitions between the largest event group, EG 2 (again characterised by mostly containing dyslipidemia and hypertension observations). Unlike previous sequence clusters from other prototype solutions, sequence cluster 8 Figure 5.26 starts with a majority of the subjects part of EG 3 Figure 5.27. Event group 3 appears to be composed primarily of cardiac-related conditions. The age profile

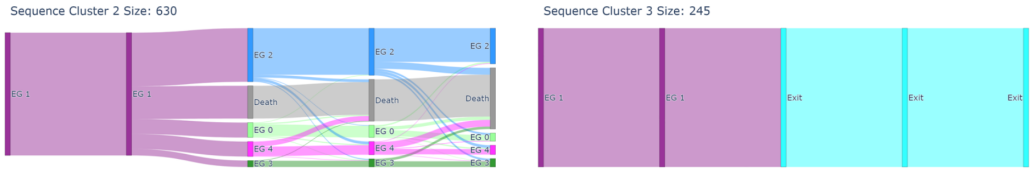

Figure 5.23: Sequence cluster 2 and 3 for Solution 550.

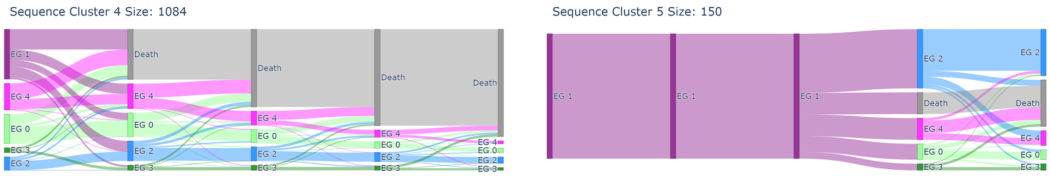

Figure 5.24: Sequence cluster 4 and 5 for Solution 550.

for this specific cluster is shown in Figure 5.28.

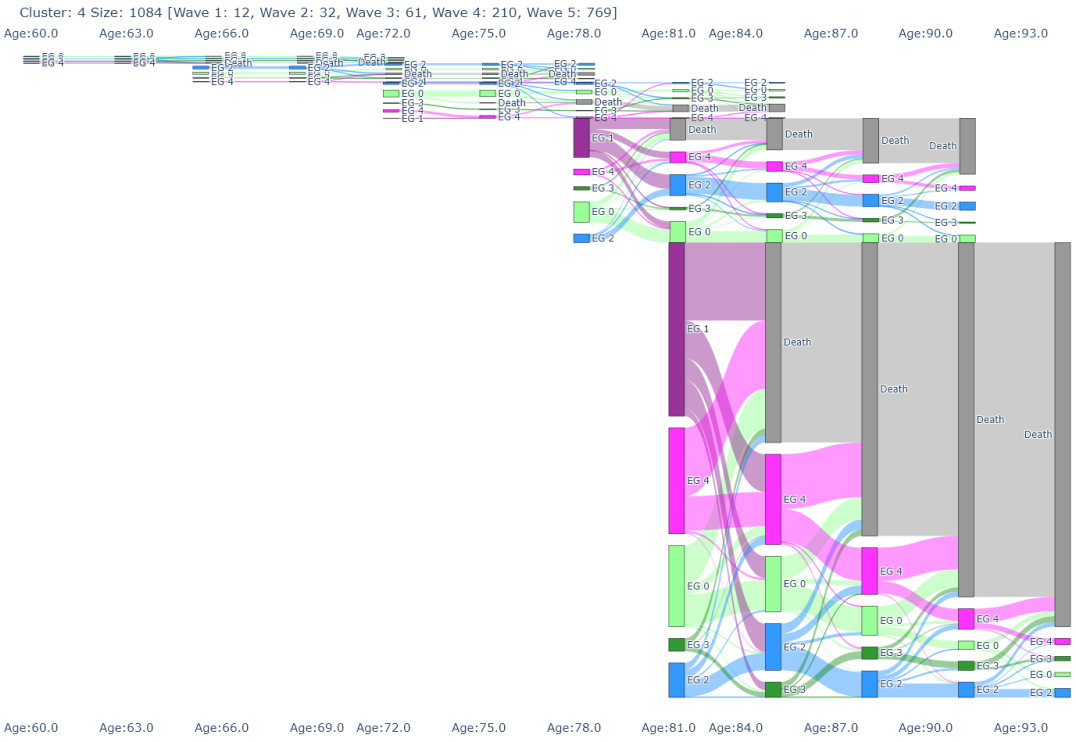

Figure 5.25: Age profile of Cluster 4 for Solution 550.

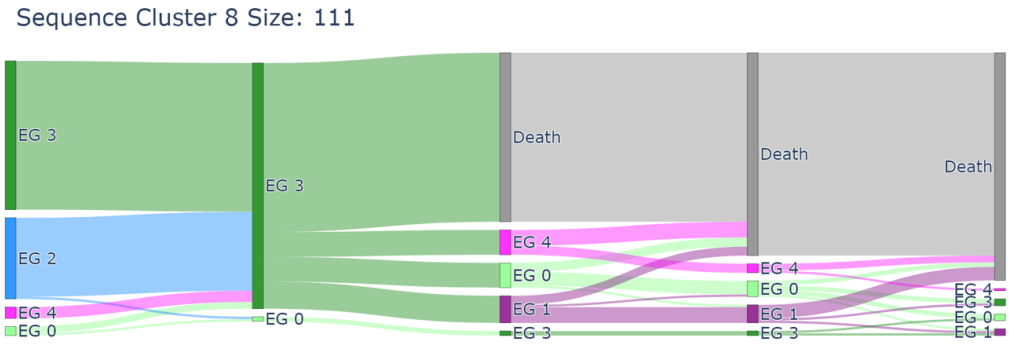

Figure 5.26: Sequence cluster 8 from Solution 51.

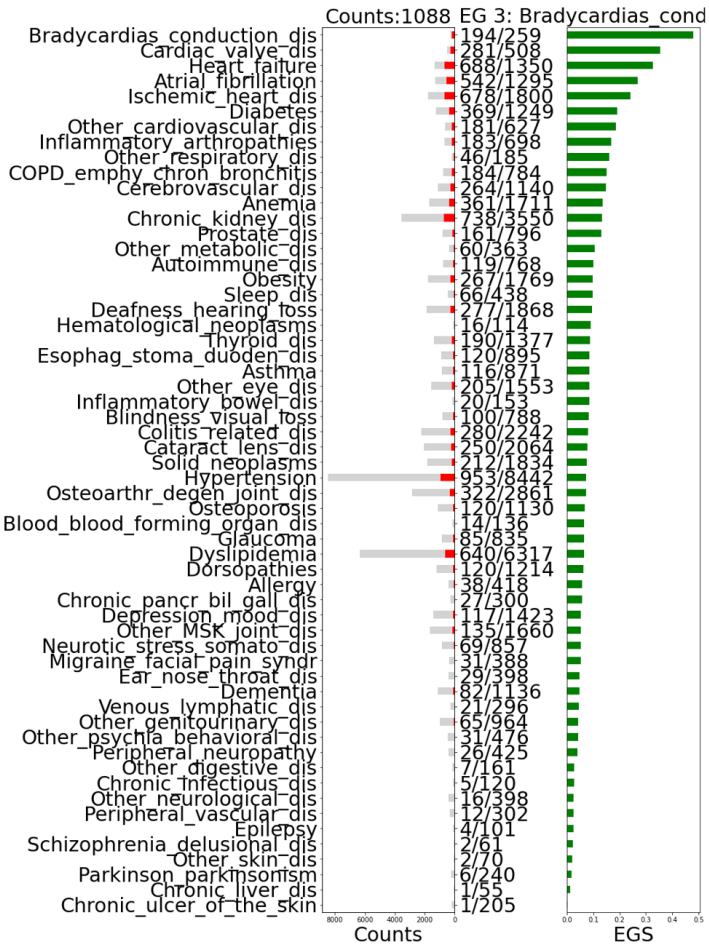

Figure 5.27: EG 3 from Solution 51.

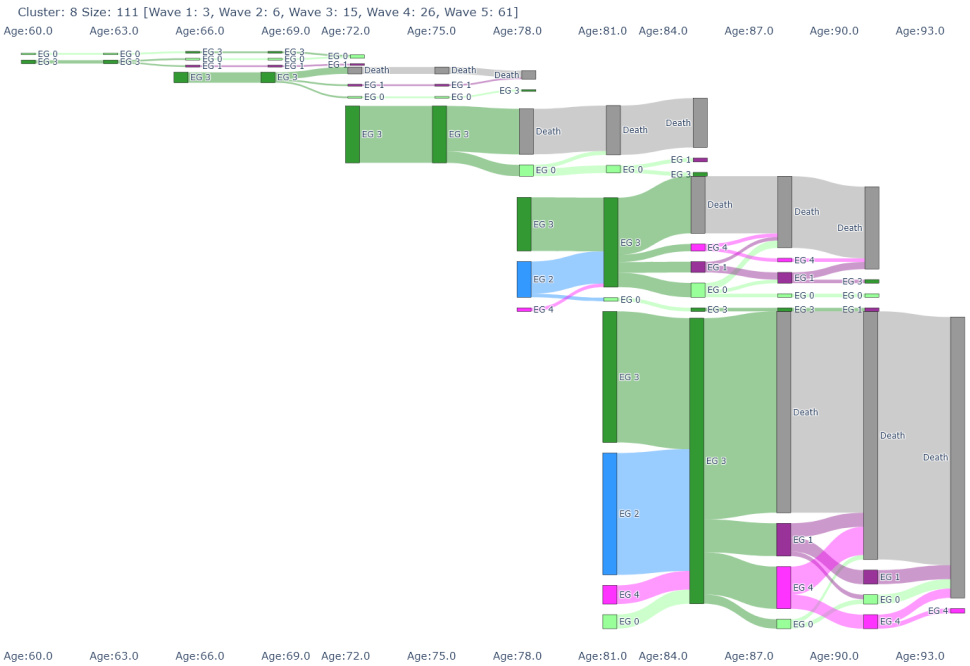

Figure 5.28: Age profile of Sequence cluster 8 from Solution 51.

5.5 CARE75+ ClusterView Analysis

The total count of solutions after the hyperparameter optimisation was 2000. Selecting the 5^{th} percentile of loss values, meant that 100 solutions were selected, and a threshold of 0.8 was used. The full parameters are shown in Figure 5.29. The graph of solutions is shown in Figure 5.30.

Figure 5.29: Parameters of the CARE75+ in the ClusterView GUI

5.5.1 CARE75+ Prototype solutions

1940

Solution 1940 has 4 total Event groups (2 of these are displayed in Figure 5.33, while the rest can be found in the online appendix), and 10 subject clusters. Similar to the EGs from the SNAC-k results, there is also a large EG which contains occurrences of Hypothyroidism, Osteoarthritis and the most common condition in the data: Hypertension. Subject cluster 1 contains only transitions between EG 1, while Clusters 7,8 and 9 start in EG 1, before progressing to different EGs. Sequence clusters 3, 4 and 5 Figure 5.35, Figure 5.36 mostly contain transitions between EG 4 Figure 5.34, which contains various conditions, and a large portion related to cataract conditions.

1947

Solution 1947 is composed of 7 EGs and 10 sequence clusters. Once again, the largest Event Group (EG 1) is the one containing the Hypothyroidism and Hypertension Figure 5.39. There is a large variety of different conditions represented in the 7 Event groups. EG 0 is mostly composed of cataract-related conditions, while the most prominent conditions in EG 2 and 6 are cancer-related. The sequence clusters again most often start with EG 1 and later progress to other event groups, such as sequence clusters 1, 6, 8 and 9.

1569

Solution 1569 is quite different from the rest of the prototype solutions. Only containing 4 sequence clusters and 8 Event groups. The sequence clusters in this solution are also distinct from other prototypes Figure 5.40. Sequence clusters 0 and 3 are familiar to previous solutions, as there is a large proportion of subjects in EG 7, which is the largest EG characterised by hypothyroidism and hypertension. Clusters 1 and 2 are rather dissimilar to any previously encountered subject cluster, as the trajectory of EG is quite varied. It appears that due to the low resolution, the resulting subject clusters combined previously separated subjects into a single cluster. This splitting of clusters in other solutions can be seen in the solution sankey diagram Figure 5.32.

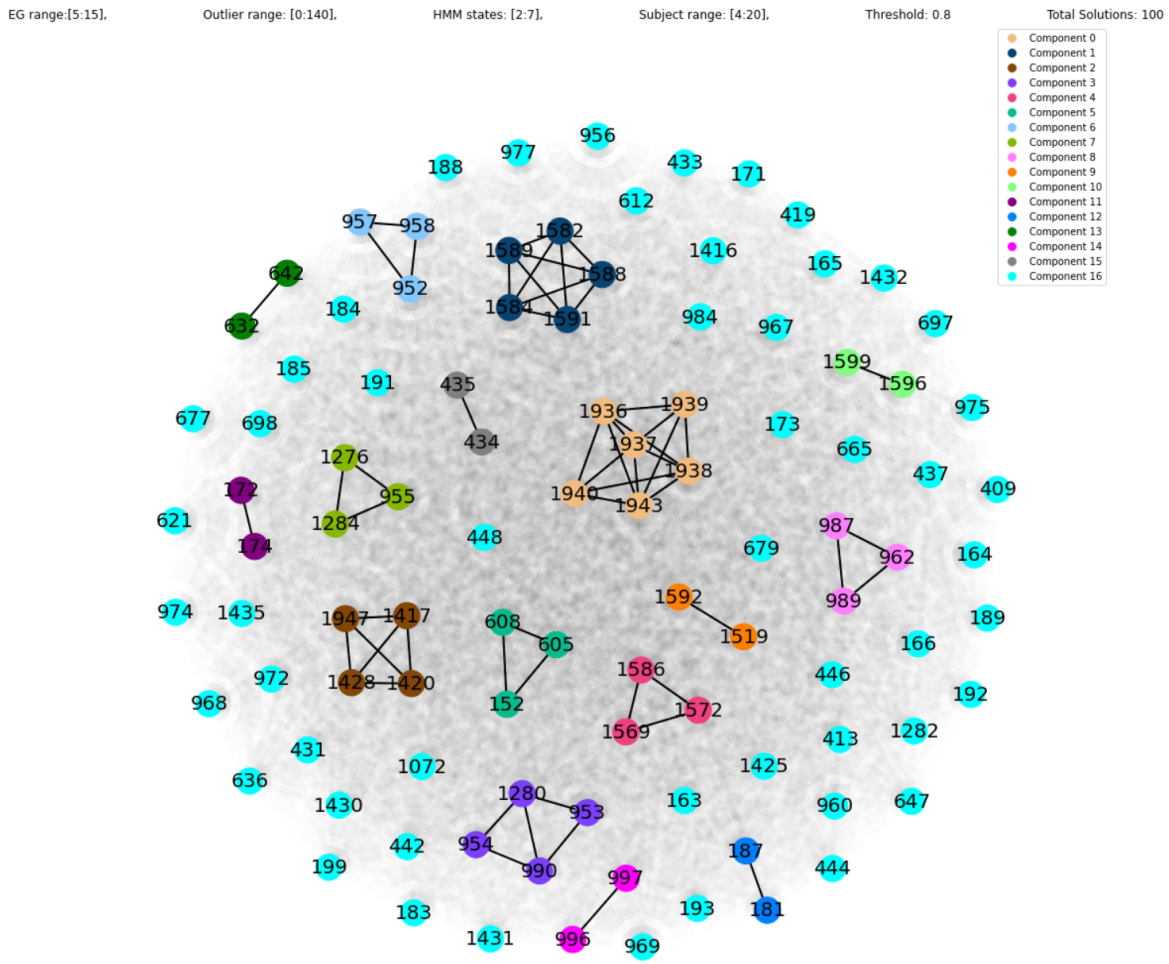

Figure 5.30: Graph of the CARE75+ in the ClusterView GUI

152

Similar to the previous Solution 1569, Solution 152 also has a low number of sequence clusters - 5, and 6 Event groups. Again, as in previous solutions, sequence clusters 0, 4 and 3 mostly contain transitions between the hypothyroidism and hypertension event group (EG 1) Figure 5.41. This prototype solution also has a low resolution of the subject clusters and groups subjects together that in other prototype solutions are in separate sequence clusters.

5.6 Discussion

In addition to identifying similarities, the ClusterView GUI allowed for a detailed comparison of clustering solutions across both datasets. By leveraging the similarity graph and examining prototype solutions, we could reduce the redundancy inherent in the clustering solutions and focus on unique and meaningful results. The presence of multiple equivalent solutions during hyperparameter optimisation, with slight variations in clustering parameters, demonstrated

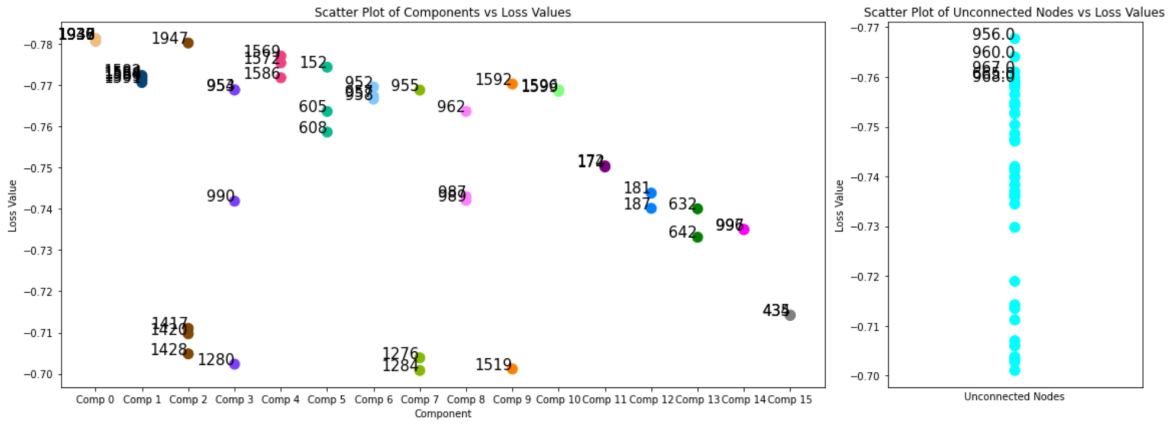

Figure 5.31: Loss values per component of the CARE75+ in the ClusterView GUI

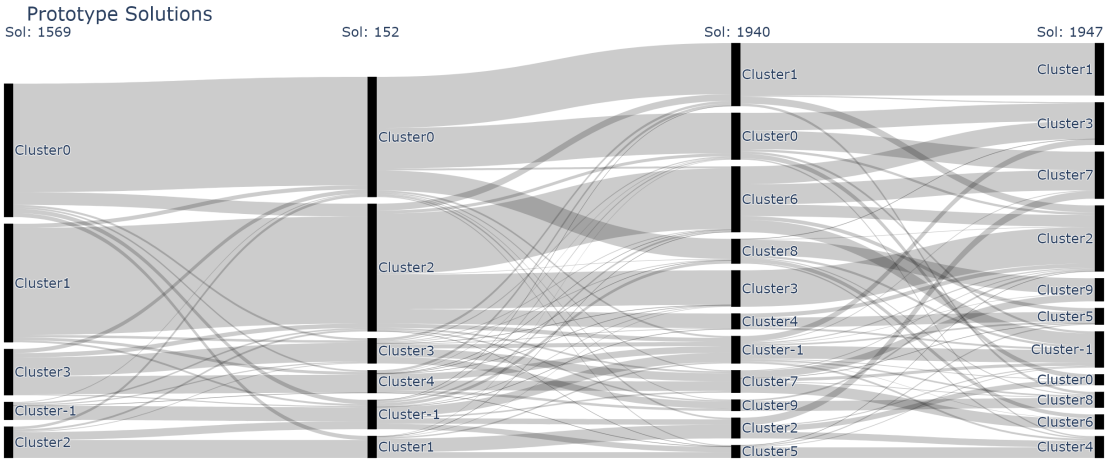

Figure 5.32: Sankey solution diagram of the 4 prototypes from the CARE75+ dataset.

the importance of systematically analysing these solutions to extract the most representative ones.

Moreover, comparing Event Groups and subject clusters across different components using Sankey diagrams provided a powerful way to visualise the relationships between different solutions. In both datasets, there were consistently stable event groups, such as those primarily consisting of dyslipidemia and hypertension. These stable clusters could indicate common early signs of multimorbidity, while variations in more complex clusters reflected the divergence in health outcomes among different individuals. This type of analysis helps in understanding the different stages of chronic disease progression and identifying which factors may contribute to a patient's transition from one health state to another.

The analysis also showed that certain event groups are highly dependent on the age distribution of subjects. The SNAC-k dataset indicated that younger subjects tended to remain in the less complicated clusters, such as the event group comprising dyslipidemia and

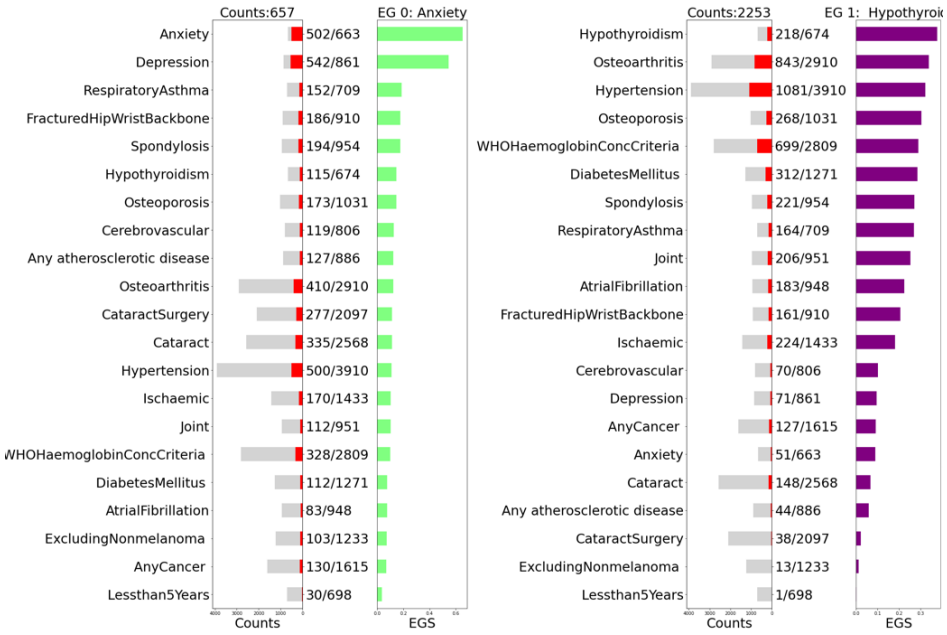

Figure 5.33: EG 0 and 1 from Solution 1940.

hypertension, while older cohorts were more likely to transition into more varied and complex event groups. Such findings provide insights into how age plays a critical role in the evolution of multimorbidity and highlight the need for tailored interventions at different stages of life.

By analysing the prototype solutions from both datasets, we observed variations in cluster sizes and the number of event groups between different solutions. This variability highlights the complexity of multimorbidity clustering and the need for a flexible approach to explore different solutions, each representing different aspects of multimorbidity progression. Understanding these variations provides a more comprehensive understanding of multimorbidity, allowing for the identification of common patterns while still accounting for unique individual trajectories.

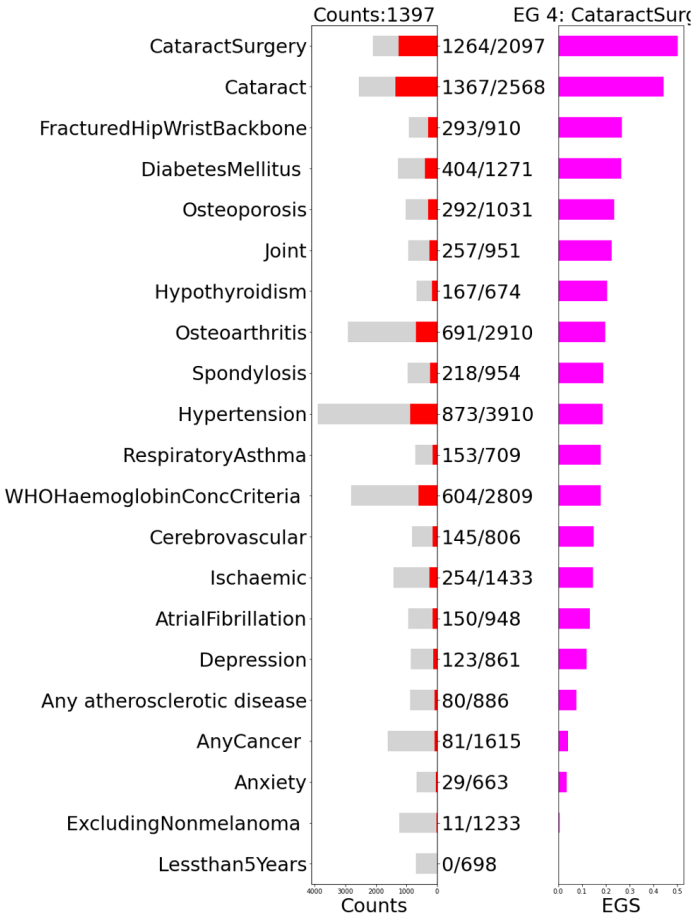

Figure 5.34: EG 4 from Solution 1940.

Figure 5.35: Sequence cluster 3 from solution 1940.

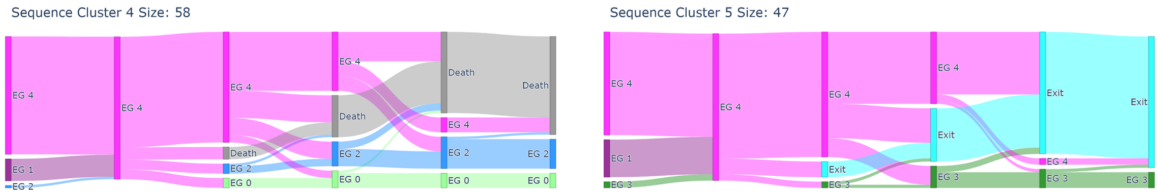

Figure 5.36: Sequence cluster 4 and 5 from solution 1940.

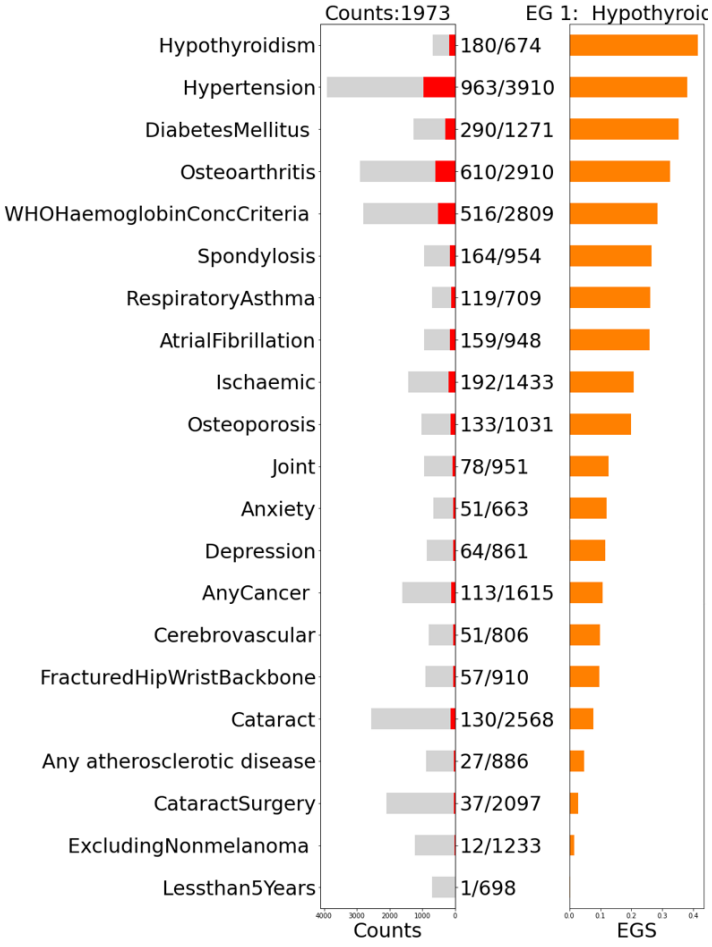

Figure 5.37: EG 1 from solution 1947.

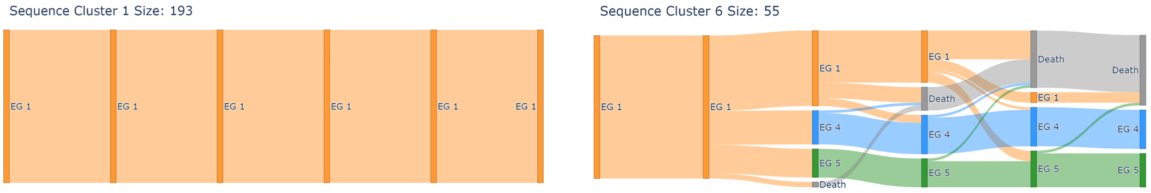

Figure 5.38: Sequence clusters 1 and 6 from solution 1947.

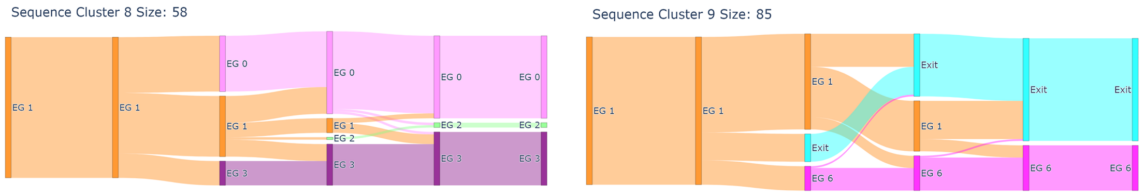

Figure 5.39: Sequence Clusters 8 and 9 from solution 1947.

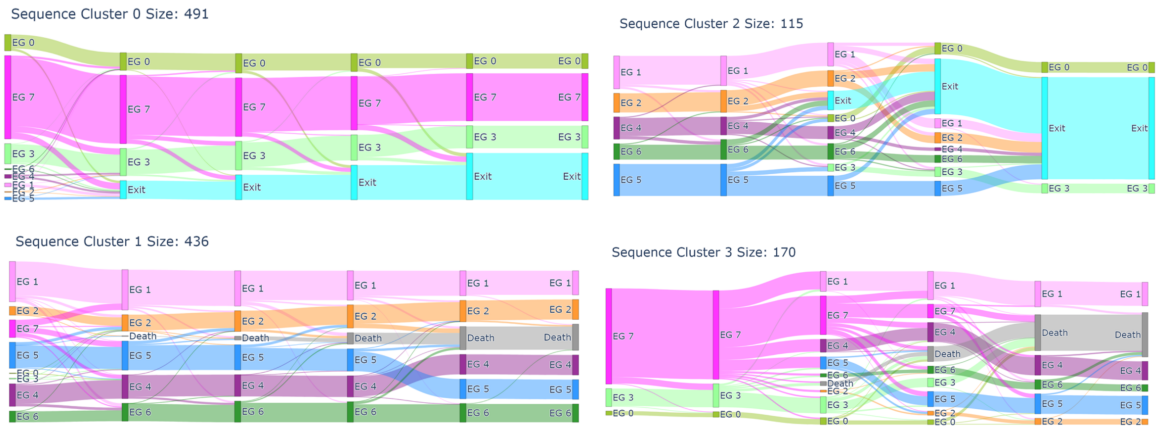

Figure 5.40: Sequence Clusters 0, 1, 2 and 3 from solution 1569.

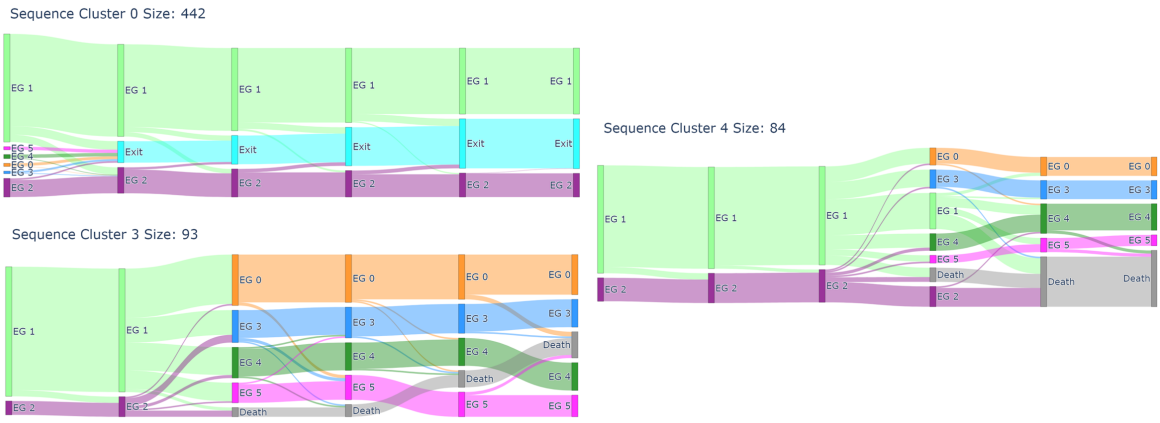

Figure 5.41: Sequence Clusters 0, 4 and 3 from solution 152.

Chapter 6

Conclusions

The primary goal of this thesis was to develop and apply a novel methodology for clustering multivariate time-series data, with a focus on analysing medical datasets to extract meaningful patterns and insights. The main contributions of this thesis are:

1. **Sequence Clustering Pipeline for Multivariate Data**: In Chapter 3, we introduced the sequence clustering pipeline designed to cluster multivariate time-series data in two stages. By employing an approach that first reduces the dimensionality of the data and then applies clustering using dynamic distances, we effectively captured the underlying temporal dependencies within the sequences. This two-stage method allowed for both a reduction in complexity and a more accurate representation of temporal relationships, facilitating more meaningful clustering outcomes.

2. **Methodology to Analyse Large Sets of Clustering Solutions**: Chapter 4 addressed the problem of multiple equivalent clustering solutions arising during hyperparameter optimisation. We introduced a methodology to group, visualise, and interpret these solutions using a similarity-based graph approach. This approach not only reduced the number of solutions needing detailed analysis but also provided a clearer understanding of the similarities and distinctions between clustering results. Sankey diagrams were implemented to visualise clustering solutions and provide an intuitive understanding of the clustering solution landscape.

3. **ClusterView GUI for Analysing Clustering Solutions**: In Chapter 4, we also presented the ClusterView graphical user interface (GUI), which facilitates the exploration and analysis of clustering solutions. This GUI enabled users to visualise the similarities and differences between various clustering approaches, providing an accessible and user-friendly platform for selecting optimal solutions. The graph-based representation and visual tools made the methodology suitable for both technical and non-technical users, enhancing its flexibility and accessibility.

In Chapter 5, we applied the developed methodology to real-world medical datasets—the SNAC-k and CARE75+ datasets. We demonstrated how our pipeline successfully identified relevant clusters of timepoints and grouped subjects based on disease progression. This application showcased the potential of our methods to reveal hidden trends in patient health trajectories, providing new insights into the progression of chronic conditions and highlighting

the potential impact of clustering in healthcare decision-making. The prototype solutions derived from these analyses captured meaningful subgroups of patients and their health paths, emphasising the practical implications of our approach in clinical contexts.

Despite being demonstrated only on clinically focused cohorts, the pipeline is **problem–agnostic** and applicable to any multivariate time–series in which each timepoint can be represented by categorical features or by discretising continuous variables into events. This contribution distinguishes itself from existing approaches in several important ways. Unlike traditional Latent Class Analysis (LCA) methods commonly used in multimorbidity research, which assume independence between observations and impose parametric forms, our two-stage approach separates the modelling of cross-sectional co-occurrence patterns from temporal progression dynamics. This separation allows for more flexible modelling of complex temporal relationships while maintaining interpretability at both the timepoint and sequence levels. Compared to matrix factorisation and tensor decomposition methods, this approach preserves interpretability by producing event groups that correspond to distinct combinations of events and sequence clusters that map to plausible progression pathways between event groups. As presented in Chapter 5, the resulting EGs (e.g., cardiometabolic, ocular) from both clinical datasets were reproducible across cohorts and formed a compact representation of clinical states for further temporal analysis.

The second contribution addresses a critical but often overlooked problem in clustering analysis: the systematic comparison and interpretation of multiple valid clustering solutions that arise during hyperparameter optimisation. This work introduces a novel graph-based methodology that transforms the space of clustering solutions into a network structure, enabling systematic analysis of solution relationships and identification of prototype solutions. These tools have also successfully been implemented in ClusterView, presented in Section 4.4 of Chapter 4. By providing an accessible interface to complex analytical methods, the tool reduces barriers to entry for analysing and comparing clustering solutions. This addresses a significant gap in clustering methodology literature. While consensus clustering and ensemble methods exist, they typically focus on combining multiple solutions into a single result rather than systematically analysing the structure of the solution space. Existing clustering validation frameworks, such as those implemented in tools like Clustrophile-2 and VICTOR [134, 135], provide individual solution evaluation but lack systematic approaches for organising and reducing large collections of solutions. The graph-based approach provides the first comprehensive framework for mapping the landscape of clustering solutions, identifying redundant results, and focusing on meaningfully distinct alternatives. By providing a systematic way to navigate the space of clustering solutions, this framework reduces the burden on analysts and increases confidence in selected results. The identification of prototype solutions enables more efficient use of expert time and resources, as stakeholders can focus their interpretation efforts on a small number of representative solutions rather than attempting to evaluate dozens of potentially redundant alternatives. This is particularly valuable in healthcare applications, where clinical expert time is limited and the stakes of model selection are high.

The methodology's effectiveness was demonstrated on two real-world longitudinal cohorts drawn from different healthcare systems and sampling frames, providing complementary validation contexts. The Swedish National Study on Ageing and Care (SNAC-K) and the UK CARE75+ cohort captured data on older adults under distinct service arrangements, allowing an assessment of **generalisability** across populations. In both datasets, the timepoint rep-

resentation yielded **recurrent Event Groups (EGs)**. Most prominently, a cardiometabolic EG (hypertension/dyslipidaemia, often with diabetes) and an ocular EG (cataract/glaucoma and related codes), that map closely onto well-described multimorbidity motifs in older adults [5, 51, 140, 141].

At the sequence level, the HMM-based distance and DBSCAN recovered trajectory families that are consistent with prior longitudinal evidence: many individuals persist in low-complexity cardiometabolic states, while others diverge towards more complex profiles (e.g., ocular or neoplastic) over time [6, 64]. The replication of these patterns across SNAC-K and CARE75+ supports the practical utility and portability of this integrated approach.

6.1 Future work

Several avenues for future research emerge from the limitations and extensions identified in this work. Development of distributed computing approaches and approximation methods could extend the framework's applicability to population-scale healthcare databases. Integration with existing electronic health record systems and clinical workflows represents another important direction for practical implementation. Sankey diagrams in this thesis are useful as a high-level summary of aggregate flows between Event Groups (EGs), but they are not suited to tracing individual trajectories across more than two time points. Because ribbons of the sankeys necessarily bifurcate and merge, person-level paths are not identifiable, and information about ordering, recurrence is partially lost—particularly when many EGs or waves are shown. To mitigate this, the Sankeys were interpreted alongside complementary views (EG composition bar charts).

Other limitations of the method include reliance on a fixed, evenly spaced time grid and forward filling, sensitivity to distance choice (non-metric Bhattacharyya vs. metric alternatives) scalability limits due to pairwise distances and density-based clustering. Specifically, the Bhattacharyya distance can be non-metric and infinite under disjoint support, which can cause issues in further clustering. While `DBSCAN` only requires a symmetric, non-negative dissimilarity to define ε–neighbourhoods, non-metric behaviour can yield inconsistent neighbourhoods (e.g., $d(A, B)$ and $d(B, C)$ small but $d(A, C)$ large), distorting local density and complicating ε selection. A potential improvement to this would be to implement alternative metric distances.

Furthermore, the fact that a first-order Markov model can assign comparable probability to multiple orderings of the same states is problematic. In the limit of an (approximately) uniform transition matrix, sequences such as $A \to B \to C$ and $C \to B \to A$ are nearly equiprobable under the same HMM. Our distance summarises each sequence by its induced one-step transition matrix; thus, trajectories with similar transition counts (and emissions) can appear very close even if their order differs. This representation is order-insensitive beyond first-order transitions: it captures how often states follow one another, rather than the global ordering.

The ClusterView interface could be extended with additional visualisation modalities, advanced statistical testing capabilities, and integration with clinical decision support systems. Development of mobile and web-based versions could increase accessibility and adoption in diverse healthcare settings. The threshold selection process should also be considered for further improvement; a more systematic selection methodology would lead to more consistent

and robust results.

Lastly, more extensive clinical validation studies are needed to establish the clinical utility of identified patterns and their impact on patient outcomes. Extension to other healthcare domains beyond multimorbidity, such as treatment response prediction and adverse event detection, could broaden the methodology's impact.

These future directions represent natural extensions of the current work and offer opportunities to further enhance the methodology's clinical impact and practical utility.

Bibliography

1. Shivade, C. *et al.* A review of approaches to identifying patient phenotype cohorts using electronic health records. *Journal of the American Medical Informatics Association* **21,** 221–230 (2014).
2. Marengoni, A., Rizzuto, D., Wang, H.-X., Winblad, B. & Fratiglioni, L. Patterns of chronic multimorbidity in the elderly population. *Journal of the American Geriatrics Society* **57,** 225–230 (2009).
3. Vetrano, D. L. *et al.* Twelve-year clinical trajectories of multimorbidity in a population of older adults. *Nature communications* **11,** 1–9 (2020).
4. Vetrano, D. L. *et al.* Frailty and multimorbidity: a systematic review and meta-analysis. *The Journals of Gerontology: Series A* **74,** 659–666 (Apr. 2019).
5. Barnett, K. *et al.* Epidemiology of multimorbidity and implications for health care, research, and medical education: a cross-sectional study. *The Lancet* **380,** 37–43 (2012).
6. Cezard, G., McHale, C., Sullivan, F., Bowles, J. & Keenan, K. Studying trajectories of multimorbidity: a systematic scoping review of longitudinal approaches and evidence. *medRxiv,* 2020–11 (2021).
7. Xu, R. & Wunsch, D. Survey of clustering algorithms. *IEEE Transactions on neural networks* **16,** 645–678 (2005).
8. Jain, A. K. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters* **31,** 651–666 (2010).
9. MacQueen, J. B. *Some Methods for Classification and Analysis of MultiVariate Observations* in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* **1** (University of California Press, 1967), 281–297.
10. McLachlan, G. & Peel, D. *Finite Mixture Models* ISBN: 978-0-471-00626-8 (John Wiley & Sons, 2000).
11. Johnson, S. C. Hierarchical clustering schemes. *Psychometrika* **32,** 241–254 (1967).
12. Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise* in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)* **96** (1996), 226–231.
13. Bezdek, J. C., Ehrlich, R. & Full, W. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences* **10,** 191–203 (1984).

14. Tan, P.-N., Steinbach, M., Karpatne, A. & Kumar, V. *Introduction to Data Mining* 2nd. ISBN: 9780133128901 (Pearson, 2019).

15. Aggarwal, C. C., Hinneburg, A. & Keim, D. A. On the surprising behavior of distance metrics in high dimensional space. *Proceedings of the 8th International Conference on Database Theory,* 420–434 (2001).

16. Deza, M. M. & Deza, E. *Encyclopedia of Distances* (Springer, 2009).

17. Berndt, D. J. & Clifford, J. *Using dynamic time warping to find patterns in time series* in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (1994), 359–370.

18. Vlachos, M., Kollios, G. & Gunopulos, D. *Discovering similar multidimensional trajectories* in *Proceedings 18th International Conference on Data Engineering* (2002), 673–684.

19. Bagnall, A., Lines, J., Bostrom, A., Large, J. & Keogh, E. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery* **31,** 606–660 (2017).

20. Jain, A. K., Murty, M. N. & Flynn, P. J. Data clustering: a review. *ACM computing surveys (CSUR)* **31,** 264–323 (1999).

21. Sibson, R. SLINK: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal* **16,** 30–34 (1973).

22. Defays, D. An efficient algorithm for a complete link method. *The Computer Journal* **20,** 364–366 (1977).

23. Day, W. H. E. & Edelsbrunner, H. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification* **1,** 7–24 (1984).

24. Dunn, J. C. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics* **3,** 32–57 (1973).

25. Bezdek, J. C. *Pattern recognition with fuzzy objective function algorithms* (Springer, 2013).

26. Rousseeuw, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* **20,** 53–65 (1987).

27. Davies, D. L. & Bouldin, D. W. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-1,** 224–227 (1979).

28. Calinski, T. & Harabasz, J. A dendrite method for cluster analysis. *Communications in Statistics* **3,** 1–27 (1974).

29. Strehl, A. & Ghosh, J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research* **3,** 583–617 (2002).

30. Fred, A. L. & Jain, A. K. Combining multiple clusterings using evidence accumulation. *IEEE transactions on pattern analysis and machine intelligence* **27,** 835–850 (2005).

31. Zhou, P., Wang, X., Du, L. & Li, X. Clustering ensemble via structured hypergraph learning. *Information Fusion* **78,** 171–179 (2022).

32. Rabiner, L. R. An introduction to hidden Markov models. *IEEE ASSP Magazine* **3,** 4–16 (1986).

33. Baum, L. E., Petrie, T., Soules, G. & Weiss, N. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics,* 1554–1563 (1966).

34. Box, G. E., Jenkins, G. M., Reinsel, G. C. & Ljung, G. *Time series analysis: forecasting and control* (Wiley, 2015).

35. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521,** 436–444 (2015).

36. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9,** 1735–1780 (1997).

37. Rabiner, L. R. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* **77,** 257–286 (1989).

38. Durbin, R., Eddy, S. R., Krogh, A. & Mitchison, G. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* (Cambridge University Press, 1998).

39. Ghahramani, Z. & Jordan, M. I. An introduction to hidden Markov models and Bayesian networks. *International journal of pattern recognition and artificial intelligence* **15,** 9–42 (2001).

40. Jurafsky, D. & Martin, J. H. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition* (Prentice Hall, 2000).

41. Bishop, C. M. *Pattern Recognition and Machine Learning* (Springer, 2006).

42. Hautamaki, V., Nykanen, P. & Franti, P. *Time-series clustering by approximate prototypes* in *2008 19th International conference on pattern recognition* (2008), 1–4.

43. Liao, T. W. Clustering of time series data—a survey. *Pattern recognition* **38,** 1857–1874 (2005).

44. Esling, P. & Agon, C. Time-series data mining. *ACM Computing Surveys (CSUR)* **45,** 1–34 (2012).

45. DESA, U. United Nations, Department of Economic and Social Affairs, Population Division.(2019). *World urbanization prospects* (2018).

46. Tinetti, M. E., Fried, T. R. & Boyd, C. M. Designing health care for the most common chronic condition—multimorbidity. *Jama* **307,** 2493–2494 (2012).

47. Nunes, B. P., Flores, T. R., Mielke, G. I., Thumé, E. & Facchini, L. A. Multimorbidity and mortality in older adults: a systematic review and meta-analysis. *Archives of gerontology and geriatrics* **67,** 130–138 (2016).

48. Ryan, A., Wallace, E., O'Hara, P. & Smith, S. M. Multimorbidity and functional decline in community-dwelling adults: a systematic review. *Health and quality of life outcomes* **13,** 1–13 (2015).

49. Wang, L., Si, L., Cocker, F., Palmer, A. J. & Sanderson, K. A systematic review of cost-of-illness studies of multimorbidity. *Applied health economics and health policy* **16,** 15–29 (2018).

50. Pathirana, T. I. & Jackson, C. A. Socioeconomic status and multimorbidity: a systematic review and meta-analysis. *Australian and New Zealand journal of public health* **42,** 186–194 (2018).

51. Violan, C. *et al.* Prevalence, determinants and patterns of multimorbidity in primary care: a systematic review of observational studies. *PloS one* **9,** e102149 (2014).

52. Fortin, M. *et al.* Lifestyle factors and multimorbidity: a cross sectional study. *BMC public health* **14,** 1–8 (2014).

53. Guisado-Clavero, M. *et al.* Multimorbidity patterns in the elderly: a prospective cohort study with cluster analysis. *BMC geriatrics* **18,** 16 (2018).

54. Violán, C. *et al.* Multimorbidity patterns with K-means nonhierarchical cluster analysis. *BMC family practice* **19,** 108 (2018).

55. Marengoni, A. *et al.* Patterns of multimorbidity in a population-based cohort of older people: sociodemographic, lifestyle, clinical, and functional differences. *The Journals of Gerontology: Series A* **75,** 798–805 (2020).

56. Violán, C. *et al.* Soft clustering using real-world data for the identification of multimorbidity patterns in an elderly population: cross-sectional study in a Mediterranean population. *BMJ open* **9,** e029594 (2019).

57. Marengoni, A. *et al.* Comparison of disease clusters in two elderly populations hospitalized in 2008 and 2010. *Gerontology* **59,** 307–315 (2013).

58. Haug, N. *et al.* High-risk multimorbidity patterns on the road to cardiovascular mortality. *BMC medicine* **18,** 1–12 (2020).

59. Hanson, H. A., Smith, K. R. & Zimmer, Z. Reproductive history and later-life comorbidity trajectories: A Medicare-linked cohort study from the Utah Population Database. *Demography* **52,** 2021–2049 (2015).

60. Jackson, C. A., Dobson, A., Tooth, L. & Mishra, G. D. Body mass index and socioeconomic position are associated with 9-year trajectories of multimorbidity: a population-based study. *Preventive medicine* **81,** 92–98 (2015).

61. Hiyoshi, A., Fall, K., Bergh, C. & Montgomery, S. Comorbidity trajectories in working age cancer survivors: a national study of Swedish men. *Cancer epidemiology* **48,** 48–55 (2017).

62. Pugh, M. J. *et al.* A retrospective cohort study of comorbidity trajectories associated with traumatic brain injury in veterans of the Iraq and Afghanistan wars. *Brain injury* **30,** 1481–1490 (2016).

63. Jung, T. & Wickrama, K. A. An introduction to latent class growth analysis and growth mixture modeling. *Social and personality psychology compass* **2,** 302–317 (2008).

64. Strauss, V. Y., Jones, P. W., Kadam, U. T. & Jordan, K. P. Distinct trajectories of multimorbidity in primary care were identified using latent class growth analysis. *Journal of clinical epidemiology* **67,** 1163–1171 (2014).

65. Martin-Sanchez, F. & Verspoor, K. Big data in medicine is driving big changes. *Yearbook of medical informatics* **9,** 14 (2014).

66. Yadav, P., Steinbach, M., Kumar, V. & Simon, G. Mining Electronic Health Records (EHRs) A Survey. *ACM Computing Surveys (CSUR)* **50,** 1–40 (2018).

67. Demner-Fushman, D., Chapman, W. W. & McDonald, C. J. What can natural language processing do for clinical decision support? *Journal of biomedical informatics* **42,** 760–772 (2009).

68. Hidalgo, C. A., Blumm, N., Barabási, A.-L. & Christakis, N. A. A dynamic network approach for the study of human phenotypes. *PLoS Comput Biol* **5,** e1000353 (2009).

69. Jensen, A. B. *et al.* Temporal disease trajectories condensed from population-wide registry data covering 6.2 million patients. *Nature communications* **5,** 1–10 (2014).

70. Giannoula, A., Gutierrez-Sacristán, A., Bravo, Á., Sanz, F. & Furlong, L. I. Identifying temporal patterns in patient disease trajectories using dynamic time warping: A population-based study. *Scientific reports* **8,** 1–14 (2018).

71. Giannoula, A., Centeno, E., Mayer, M.-A., Sanz, F. & Furlong, L. I. A system-level analysis of patient disease trajectories based on clinical, phenotypic and molecular similarities. *Bioinformatics* (2020).

72. Aguado, A., Moratalla-Navarro, F., López-Simarro, F. & Moreno, V. Morbinet: multimorbidity networks in adult general population. Analysis of type 2 diabetes mellitus comorbidity. *Scientific reports* **10,** 1–12 (2020).

73. Hassaine, A., Salimi-Khorshidi, G., Canoy, D. & Rahimi, K. Untangling the complexity of multimorbidity with machine learning. *Mechanisms of Ageing and Development* **190,** 111325 (2020).

74. Hripcsak, G. & Albers, D. J. Next-generation phenotyping of electronic health records. *Journal of the American Medical Informatics Association* **20,** 117–121 (2013).

75. Ho, J. C. *et al.* Limestone: High-throughput candidate phenotype generation via tensor factorization. *Journal of biomedical informatics* **52,** 199–211 (2014).

76. Wang, Y. *et al. Rubik: Knowledge guided tensor factorization and completion for health data analytics* in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015), 1265–1274.

77. Henderson, J. *et al. Granite: Diversified, sparse tensor factorization for electronic health record-based phenotyping* in *2017 IEEE international conference on healthcare informatics (ICHI)* (2017), 214–223.

78. Zhou, J., Wang, F., Hu, J. & Ye, J. *From micro to macro: data driven phenotyping by densification of longitudinal electronic medical records* in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), 135–144.

79. Perros, I., Papalexakis, E. E., Vuduc, R., Searles, E. & Sun, J. Temporal phenotyping of medically complex children via PARAFAC2 tensor factorization. *Journal of biomedical informatics* **93,** 103125 (2019).

80. Zhao, J. *et al.* Detecting time-evolving phenotypic topics via tensor factorization on electronic health records: Cardiovascular disease case study. *Journal of biomedical informatics* **98,** 103270 (2019).

81. Afshar, A. *et al. TASTE: temporal and static tensor factorization for phenotyping electronic health records* in *Proceedings of the ACM Conference on Health, Inference, and Learning* (2020), 193–203.

82. Hassaine, A. *et al.* Learning multimorbidity patterns from electronic health records using non-negative matrix factorisation. *Journal of Biomedical Informatics,* 103606 (2020).

83. Chen, I. Y., Joshi, S., Ghassemi, M. & Ranganath, R. Probabilistic Machine Learning for Healthcare. *arXiv preprint arXiv:2009.11087* (2020).

84. Zhao, Y. & Udell, M. *Missing value imputation for mixed data via gaussian copula* in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020), 636–646.

85. Miscouridou, X., Perotte, A., Elhadad, N. & Ranganath, R. *Deep survival analysis: Nonparametrics and missingness* in *Machine Learning for Healthcare Conference* (2018), 244–256.

86. Pivovarov, R. *et al.* Learning probabilistic phenotypes from heterogeneous EHR data. *Journal of biomedical informatics* **58,** 156–165 (2015).

87. Mayhew, M. B. *et al.* Flexible, cluster-based analysis of the electronic medical record of sepsis with composite mixture models. *Journal of biomedical informatics* **78,** 33–42 (2018).

88. Marlin, B. M., Kale, D. C., Khemani, R. G. & Wetzel, R. C. *Unsupervised pattern discovery in electronic health care data using probabilistic clustering models* in *Proceedings of the 2nd ACM SIGHIT international health informatics symposium* (2012), 389–398.

89. Li, Y. *et al.* Inferring multimodal latent topics from electronic health records. *Nature communications* **11,** 1–17 (2020).

90. Blei, D. M., Ng, A. Y. & Jordan, M. I. Latent dirichlet allocation. *the Journal of machine Learning research* **3,** 993–1022 (2003).

91. Taherdoost, H. Deep learning and neural networks: challenges and applications. *Symmetry* **15.** Highlights the requirement for large, diverse datasets to enable generalization and avoid overfitting. (2023).

92. Alzubaidi, L. *et al.* A survey on deep learning tools dealing with data scarcity. *Journal of Big Data.* Surveys challenges of limited data in DL and reviews strategies like transfer learning and data augmentation. (2023).

93. Salman, S. & Liu, X. Overfitting Mechanism and Avoidance in Deep Neural Networks. *arXiv preprint arXiv:1901.06566.* Analyzes fundamental overfitting mechanisms in deep neural networks, especially under limited sample sizes. (2019).

94. Violán, C. *et al.* Five-year trajectories of multimorbidity patterns in an elderly Mediterranean population using Hidden Markov Models. *Scientific reports* **10,** 1–11 (2020).

95. Shi, X. *et al.* Development of multimorbidity over time: an analysis of Belgium primary care data using Markov chains and Weighted Association Rules Mining. *The Journals of Gerontology: Series A* (2020).

96. Galagali, N. & Xu-Wilson, M. Patient Subtyping with Disease Progression and Irregular Observation Trajectories. *arXiv preprint arXiv:1810.09043* (2018).

97. Huang, Z., Dong, W., Wang, F. & Duan, H. *Medical inpatient journey modeling and clustering: a Bayesian hidden Markov model based approach* in *AMIA Annual Symposium Proceedings* **2015** (2015), 649.

98. Ghassempour, S., Girosi, F. & Maeder, A. Clustering multivariate time series using hidden Markov models. *International journal of environmental research and public health* **11,** 2741–2763 (2014).

99. Zhang, Y. & Padman, R. Innovations in chronic care delivery using data-driven clinical pathways. *Am J Manag Care* **21,** e661–e668 (2015).

100. Li-wei, H. L. *et al.* A physiological time series dynamics-based approach to patient monitoring and outcome prediction. *IEEE journal of biomedical and health informatics* **19,** 1068–1076 (2014).

101. Li-wei, H. L. *et al. Tracking progression of patient state of health in critical care using inferred shared dynamics in physiological time series* in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2013), 7072–7075.

102. Li-wei, H. L., Nemati, S., Moody, G. B., Heldt, T. & Mark, R. G. *Uncovering clinical significance of vital sign dynamics in critical care* in *Computing in Cardiology 2014* (2014), 1141–1144.

103. Yao, Y., Zhao, X., Wu, Y., Zhang, Y. & Rong, J. Clustering driver behavior using dynamic time warping and hidden Markov model. *Journal of Intelligent Transportation Systems* **25,** 249–262 (2021).

104. Liu, Y.-Y., Li, S., Li, F., Song, L. & Rehg, J. M. Efficient learning of continuous-time hidden markov models for disease progression. *Advances in neural information processing systems* **28,** 3599 (2015).

105. Maag, B., Feuerriegel, S., Kraus, M., Saar-Tsechansky, M. & Züger, T. *Modeling longitudinal dynamics of comorbidities* in *Proceedings of the Conference on Health, Inference, and Learning* (2021), 222–235.

106. Alaa, A. & van der Schaar, M. Attentive state-space modeling of disease progression. *Advances in Neural Information Processing Systems* **33** (2019).

107. Kriegel, H.-P., Kröger, P. & Zimek, A. *Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering* in *ACM Transactions on Knowledge Discovery from Data (TKDD)* **3** (ACM, 2009), 1–58.

108. Le Roux, B. & Rouanet, H. *Geometric data analysis: from correspondence analysis to structured data analysis* (Springer Science & Business Media, 2004).

109. García-García, D., Parrado-Hernández, E. & Diaz-de-Maria, F. State-space dynamics distance for clustering sequential data. *Pattern Recognition* **44,** 1014–1022 (2011).

110. Smyth, P. Clustering sequences with hidden Markov models. *Advances in neural information processing systems* **9** (1996).

111. Viterbi, A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory* **13,** 260–269 (1967).

112. Bhattacharyya, A. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.* **35,** 99–109 (1943).

113. Sculley, D. *Web-scale k-means clustering* in *Proceedings of the 19th International Conference on World Wide Web (WWW)* Introduces mini-batch k-means with near-linear passes over data (2010), 1177–1178.

114. Zhang, T., Ramakrishnan, R. & Livny, M. *BIRCH: An efficient data clustering method for very large databases* in *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* CF-tree yields roughly linear time when it fits in memory; assumes Euclidean statistics (CFs) (1996), 103–114.

115. Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. *A density-based algorithm for discovering clusters in large spatial databases with noise* in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)* Contrasts density-based clustering (noise detection, arbitrary shapes) with centroid-based methods (1996), 226–231.

116. Otto, E. *et al.* Overview of Sankey flow diagrams: Focusing on symptom trajectories in older adults with advanced cancer. *Journal of geriatric oncology* **13,** 742–746 (2022).

117. Bergstra, J. & Bengio, Y. Random search for hyper-parameter optimization. *Journal of machine learning research* **13** (2012).

118. Snoek, J., Larochelle, H. & Adams, R. P. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems* **25** (2012).

119. Bergstra, J., Yamins, D. & Cox, D. *Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures* in *International conference on machine learning* (2013), 115–123.

120. Bergstra, J., Komer, B., Eliasmith, C., Yamins, D. & Cox, D. D. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery* **8,** 014008 (2015).

121. Hubert, L. & Arabie, P. Comparing partitions. *Journal of classification* **2,** 193–218 (1985).

122. Fowlkes, E. B. & Mallows, C. L. A method for comparing two hierarchical clusterings. *Journal of the American statistical association* **78,** 553–569 (1983).

123. Gates, A. J., Wood, I. B., Hetrick, W. P. & Ahn, Y.-Y. Element-centric clustering comparison unifies overlaps and hierarchy. *Scientific reports* **9,** 8574 (2019).

124. Hennig, C., Meila, M., Murtagh, F. & Rocci, R. *Handbook of cluster analysis* (CRC press, 2015).

125. Bae, J. *et al.* Interactive Clustering: A Comprehensive Review. *ACM Comput. Surv.* **53,** 1:1–1:39. ISSN: 0360-0300. `https://dl.acm.org/doi/10.1145/3340960` (Feb. 2020).

126. Von Landesberger, T. *et al. Visual analysis of large graphs: state-of-the-art and future research challenges* in *Computer graphics forum* **30** (2011), 1719–1749.

127. Turkay, C., Parulek, J., Reuter, N. & Hauser, H. *Integrating cluster formation and cluster evaluation in interactive visual analysis* in *Proceedings of the 27th Spring Conference on Computer Graphics* (Association for Computing Machinery, New York, NY, USA, Apr. 2011), 77–86. ISBN: 978-1-4503-1978-2.

128. Seo, J. & Shneiderman, B. Interactively exploring hierarchical clustering results [gene identification]. *Computer* **35,** 80–86 (2002).

129. Diehl, A., Satan, U., Halter, G., Flückiger, B. & Pajarola, R. *SnakeTrees: Multi-level Visual Exploration of High-dimensional Clustered Data* in *Diehl, Alexandra; Satan, Uensal; Halter, Gaudenz; Flückiger, Barbara; Pajarola, Renato (2024). SnakeTrees: Multi-level Visual Exploration of High-dimensional Clustered Data. In: The 1st Japan Visualisation Symposium, Tokyo, 22 April 2024 - 23 April 2024, Visualization Society of Japan.* (Visualization Society of Japan, Tokyo, Apr. 2024).

130. Lex, A., Streit, M., Partl, C., Kashofer, K. & Schmalstieg, D. Comparative Analysis of Multidimensional, Quantitative Data. *IEEE Transactions on Visualization and Computer Graphics* **16,** 1027–1035. ISSN: 1941-0506 (Nov. 2010).

131. Von Landesberger, T., Gorner, M. & Schreck, T. *Visual analysis of graphs with multiple connected components* in *2009 IEEE Symposium on Visual Analytics Science and Technology* (Oct. 2009), 155–162.

132. L'Yi, S. *et al.* XCluSim: a visual analytics tool for interactively comparing multiple clustering results of bioinformatics data. *BMC Bioinformatics* **16,** S5. ISSN: 1471-2105 (Aug. 2015).

133. Kwon, B. C. *et al.* Clustervision: Visual Supervision of Unsupervised Clustering. *IEEE Transactions on Visualization and Computer Graphics* **24,** 142–151. ISSN: 1941-0506 (Jan. 2018).

134. Cavallo, M. & Demiralp, C. Clustrophile 2: Guided Visual Clustering Analysis. *IEEE Transactions on Visualization and Computer Graphics* **25,** 267–276. ISSN: 1941-0506 (Jan. 2019).

135. Karatzas, E. *et al.* VICTOR: A visual analytics web application for comparing cluster sets. *Comput. Biol. Med.* **135.** ISSN: 0010-4825 (Aug. 2021).

136. Seo, J. & Shneiderman, B. Interactively exploring hierarchical clustering results [gene identification]. *Computer* **35,** 80–86 (2002).

137. Blondel, V. D., Guillaume, J.-L., Lambiotte, R. & Lefebvre, E. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* **2008,** P10008 (2008).

138. Calderón-Larrañaga, A. *et al.* Assessing and measuring chronic multimorbidity in the older population: a proposal for its operationalization. *Journals of Gerontology Series A: Biomedical Sciences and Medical Sciences* **72,** 1417–1423 (2017).

139. Heaven, A. *et al.* Community ageing research 75+ study (CARE75+): an experimental ageing and frailty research cohort. *BMJ open* **9,** e026744 (2019).

140. Prados-Torres, A., Calderón-Larrañaga, A., Hancco-Saavedra, J., Poblador-Plou, B. & van den Akker, M. Multimorbidity patterns: a systematic review. *Journal of clinical epidemiology* **67,** 254–266 (2014).

141. Garin, N. *et al.* Global Multimorbidity Patterns: A Cross-Sectional, Population-Based, Multi-Country Study. *The Journals of Gerontology: Series A* **71,** 205–214. https://academic.oup.com/biomedgerontology/article/71/2/205/2605626 (2016).

Activities and publications during PhD

Part of the work presented in this thesis has also been presented at the following events:

Summer School Attendance

- Oxford Machine Learning Summer School 2023: ML x Health, July 13th – July 16th 2023.

Talk and Conference Paper

- "HMM-based clustering of multimorbidity trajectories", Manchester Advances in Data Science and AI Conference 2022, June 20th and 21st.

Presentation Symposium

- "Alternative Clustering of Multivariate Health Sequences", HELSI Computer Science Symposium, May 19th 2023.

Poster Presentations

- HELSI Annual Meeting 2022, October 5th.
- HELSI Annual Meeting 2023, October 2nd.
- Computer Science Research Away Day 2022, June 16th.
- Computer Science Research Away Day 2023, June 8th.

Code used During PhD

The software developed for the methods described in Chapters 3, 4 and 5 is publicly available under `https://github.com/beniulis/plotting_graph_results` repositories.

Appendix A: Similarity Calculation

Calculating the Similarity between all solutions presented a challenge due to the large number of solutions involved. To speed up this process, we utilized the Sheffield High-Performance Computing (HPC) facility. By distributing the calculations across multiple jobs, each section of the final distance matrix was computed independently. This approach significantly accelerated the computation, allowing for an efficient analysis of the clustering solutions within a reasonable timeframe.

Appendix B: High-Performance Computing Resources

To perform the hyperparameter optimisation we utilised The University of Sheffield's High Performance Computing (HPC) clusters ShARC. The HPC cluster used in this research comprises 98 publicly available nodes, which are shared across various research groups. Each node is based on the Dell PowerEdge C6320 machine. Each node is equipped with 2 Intel Xeon E5-2630 v3 processors. Each node is equipped with 64 GB of DDR4 RAM, clocked at 1866 MHz. This amounts to 4 GB of memory per CPU core

For the hyperparameter optimisation each initialisation we used a total of 4 cores with 8GB of total memory. `https://docs.hpc.shef.ac.uk/en/latest/decommissioned/sharc/cluster_specs.html#sharc-specs&gsc.tab=0`.

Appendix C: SNAC-K and CARE 75+ Condition Counts

We ranked the frequency of diagnosed conditions across all timepoints to visualise which conditions were the most prominent in the dataset. This is shown in Figure 1.

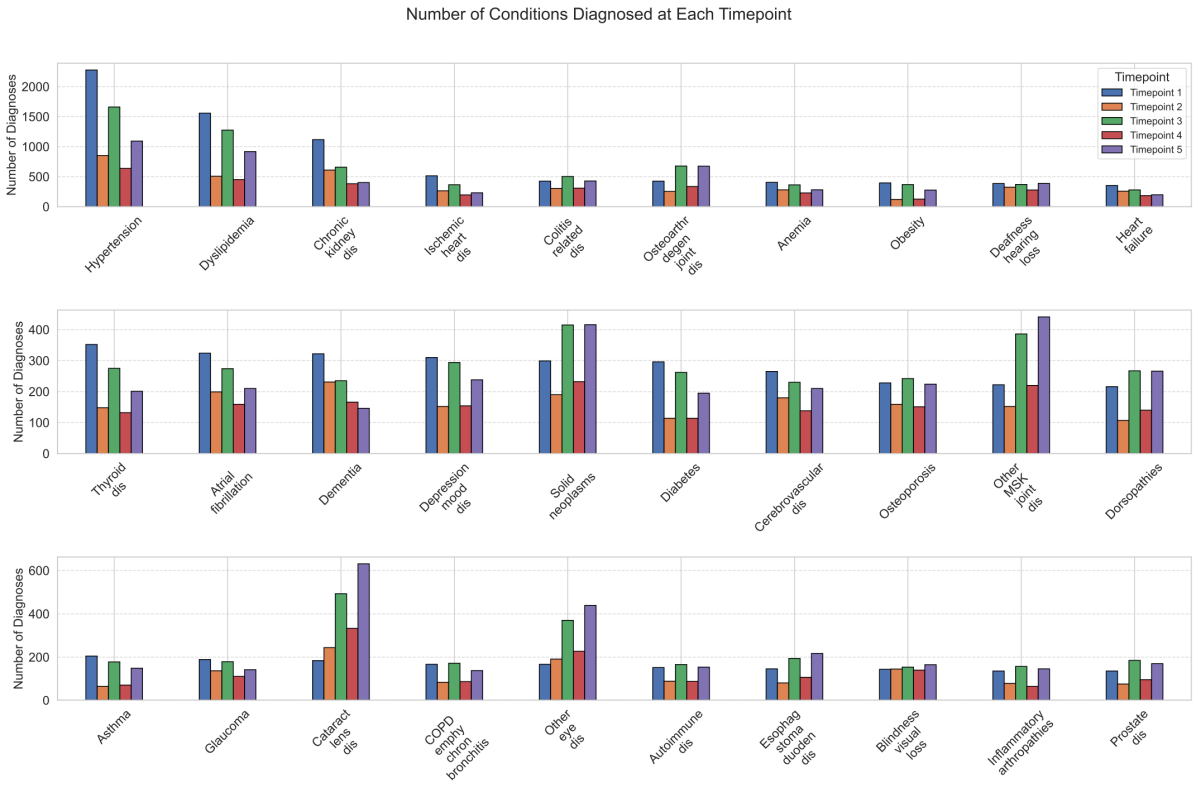

Figure 1: SNAC-k All recorded conditions per timepoint.

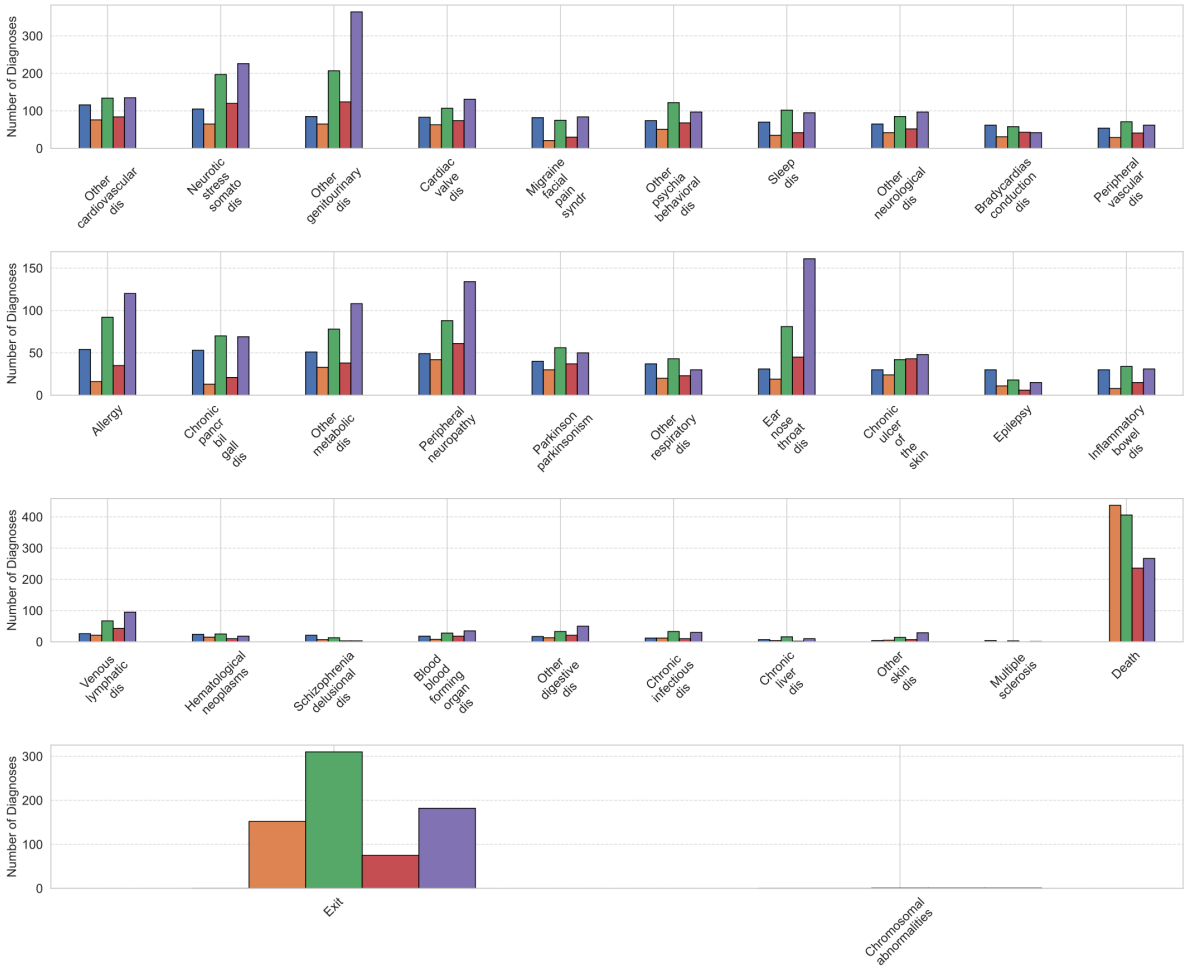

Figure 2: SNAC-k All recorded conditions per timepoint.

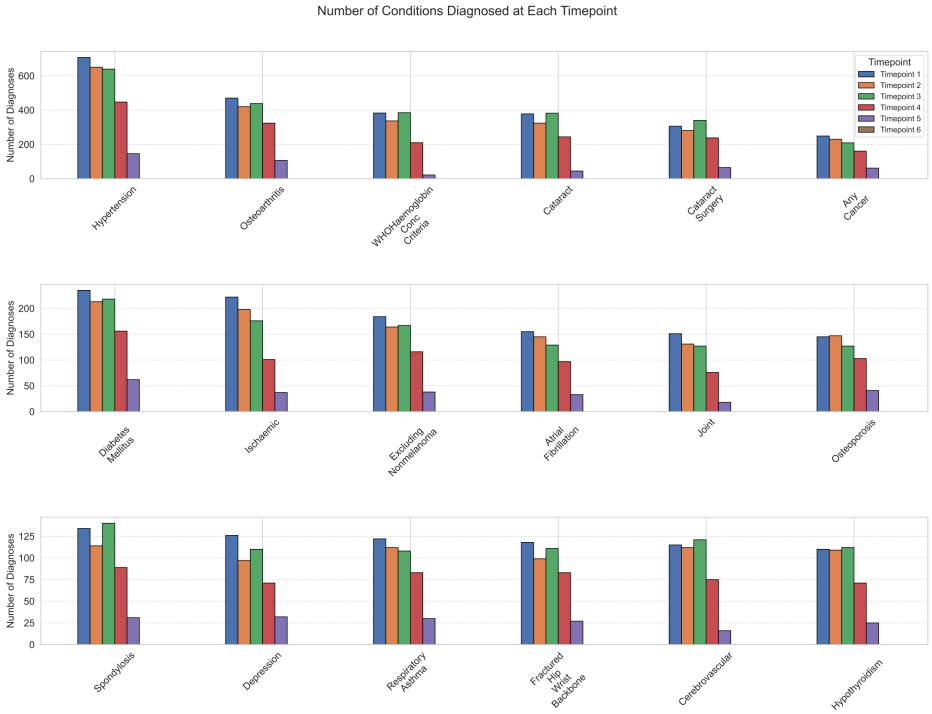

Figure 3: CARE 75+ All recorded conditions per timepoint.

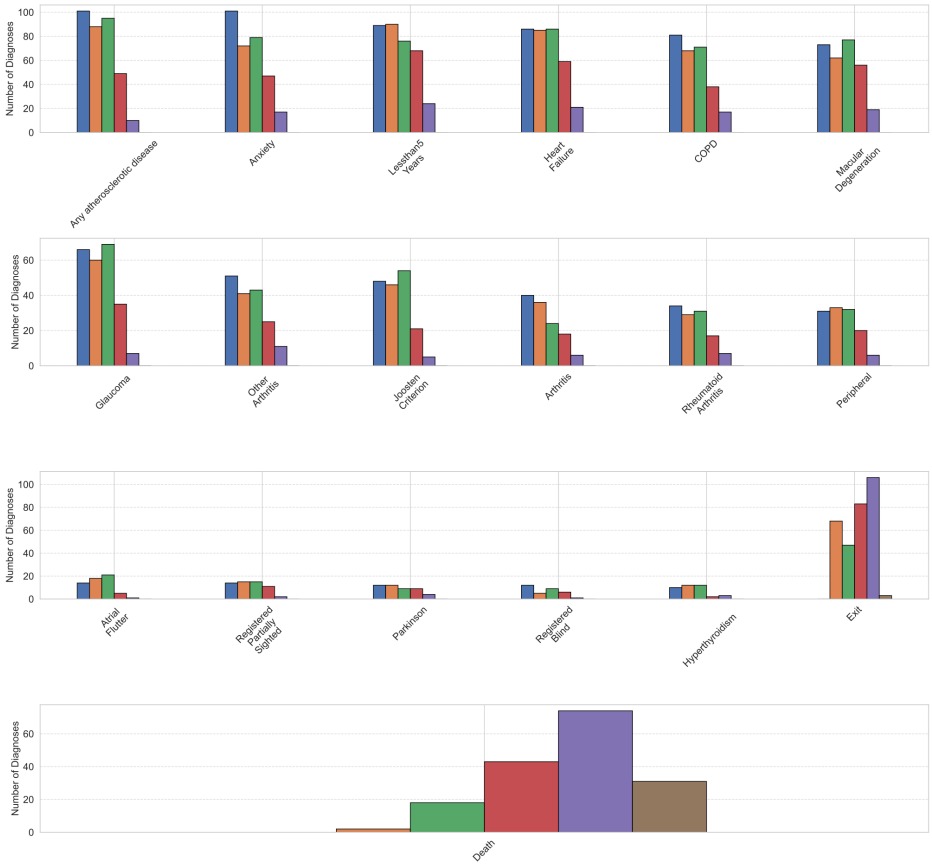

Figure 4: CARE 75+ All recorded conditions per timepoint.