4

BEYOND NETWORKS

We've explored many ways to analyze network data by measuring geometric properties. In this chapter, we'll introduce network filtration for weighted networks, which tracks geometric properties and network metrics over threshold values imposed on the network. Then we'll examine how network data can be transformed into a higher-dimensional topological object called a *simplicial complex*, and we'll explore higher-dimensional versions of the network metrics we've previously considered. From there, we'll return to graph comparisons using a tool from topology related to filtrations.

Graph Filtration

In the previous chapters, we reviewed different network metrics, including different measures of centrality, entropy, spectral radius, diameter, and many others. There's an interesting way to understand topological properties of weighted networks: *graph filtration*, a method of creating a series of weighted networks by iteratively removing edges below a certain threshold (for instance, all edges with weights lower than 0.2, 0.4, or 0.6). By creating a series of thresholded graphs, it's possible to identify persistent network metrics, or local and global network metrics that persist across a wide range of filtration values. This gives us features that can be plotted or tracked across filtrations. This is one of the core ideas of topological data analysis (TDA).

To explore this further, let's say we're examining longitudinal educational or risk behavior outcomes of adolescents based on adolescent friendship or informal social ties within a community. Imagine we have weighted social networks with high degree metrics for each vertex, where edges are weighted by hours spent with friends over a normal week. The first group of friends might spend a couple hours together playing soccer on the weekend. The second group might study together once or twice a week and see each other in classes. The third group might play sports often, do homework together after dinner or in the mornings before school, and stay over at each other's homes often. As we filter hours spent together, the degree metrics will drop for the first two groups of friends in a network. The last group will retain a high degree metric over the filtration, as they spend more time together. This persistence of degree will likely shed light on the strength of whatever social ties we're examining in our study.

Let's examine how we can implement graph filtrations by decomposing and exploring two small example social networks, Graph 1 and Graph 2. First, we'll load the two networks into R and explore the structures of the full networks with the script in Listing 4-1.

```
#load both networks in R
mydata1<-as.matrix(read.csv("Graph1w.csv",header=F))
mydata2<-as.matrix(read.csv("Graph2w.csv",header=F))

#load igraph and convert to graph objects
library(igraph)
g1<-graph_from_adjacency_matrix(mydata1,mode="undirected",weighted=T)
g2<-graph_from_adjacency_matrix(mydata2,mode="undirected",weighted=T)

#plot the two graphs
plot(g1,edge.label=E(g1)$weight,main="Graph 1")
plot(g2,edge.label=E(g2)$weight,main="Graph 2")
```

Listing 4-1: A script that loads two different network structures for filtration

The script in Listing 4-1 should load two different networks, Graph 1 and Graph 2, which have different connectivity patterns but the same number of vertices. It should also plot both networks with edge weights given in the plots. Let's compare the networks, shown in Figure 4-1.

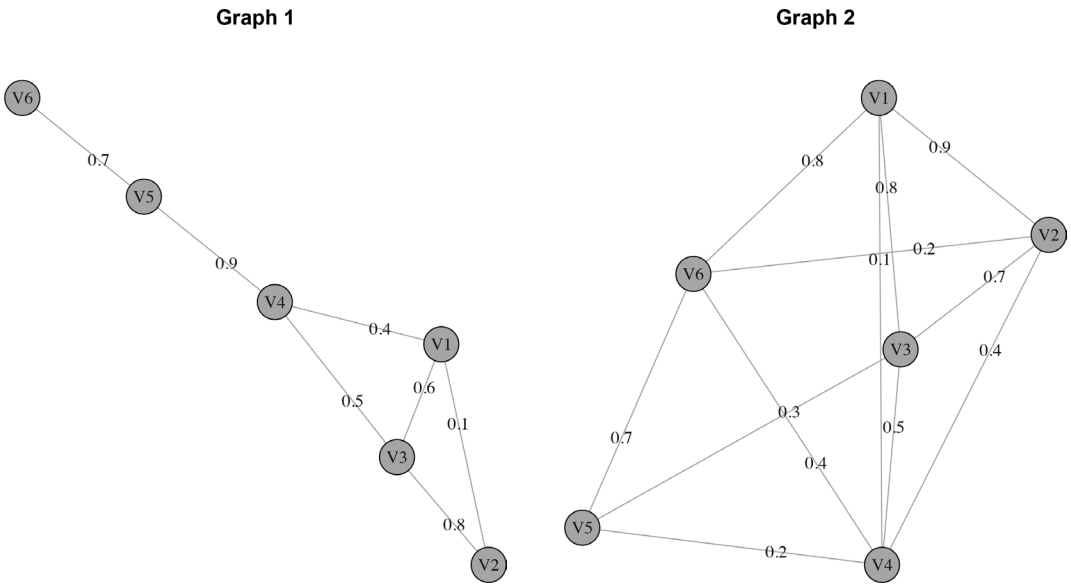

Figure 4-1: Plots of the two example networks

Figure 4-1 suggests that Graph 1 is a sparsely connected network with mostly large edge weights (perhaps a sample of students in the same class showing up for a service activity over the course of a weekend), whereas Graph 2 is a densely connected network with a mixture of different edge weights (perhaps a friendship network within a sports team). We'd expect higher hub scores and other centrality measures in Graph 2, but a filtration might change those metrics more quickly than we'd expect them to change in Graph 1.

Let's create filtrations of the networks; this will allow us to explore a few centrality metrics on these networks. We can do this by adding the following code to the script in Listing 4-1:

```
#filter Graph 1
mydata1[mydata1<0.2]<-0
g12<-graph_from_adjacency_matrix(mydata1,mode="undirected",weighted=T)
mydata1[mydata1<0.4]<-0
g14<-graph_from_adjacency_matrix(mydata1,mode="undirected",weighted=T)
mydata1[mydata1<0.6]<-0
g16<-graph_from_adjacency_matrix(mydata1,mode="undirected",weighted=T)
mydata1[mydata1<0.8]<-0
g18<-graph_from_adjacency_matrix(mydata1,mode="undirected",weighted=T)

#filter Graph 2
mydata2[mydata2<0.2]<-0
g22<-graph_from_adjacency_matrix(mydata2,mode="undirected",weighted=T)
mydata2[mydata2<0.4]<-0
g24<-graph_from_adjacency_matrix(mydata2,mode="undirected",weighted=T)
mydata2[mydata2<0.6]<-0
g26<-graph_from_adjacency_matrix(mydata2,mode="undirected",weighted=T)
mydata2[mydata2<0.8]<-0
g28<-graph_from_adjacency_matrix(mydata2,mode="undirected",weighted=T)
```

The previous code filters Graph 1 and Graph 2 by edge weight, using increasing intervals of 0.2. This yields a series of five networks in each graph filtration, which can be further examined by applying network metrics to each sequence of filtered graphs.

Let's examine the degree centrality of each vertex across the filtration of Graph 1 by adding the following to our script:

```
#calculate degree centrality for Graph 1's filtration sequence
d1<-degree(g1)
d12<-degree(g12)
d14<-degree(g14)
d16<-degree(g16)
d18<-degree(g18)

#create a dataset tracking degree centrality across the filtration
g1deg<-cbind(d1,d12,d14,d16,d18)
```

This code calculates degree centrality across filtrations of Graph 1, which should yield a dataset containing the information in Table 4-1.

Table 4-1: Degree Centrality Across Graph 1 Filtrations

Column1	d1	d12	d14	d16	d18
V1	3	2	2	1	0
V2	2	1	1	1	1
V3	3	3	3	2	1
V4	3	3	3	1	1
V5	2	2	2	2	1
V6	1	1	1	1	0

Table 4-1 shows that vertices 1, 3, and 4 have high degree centralities; however, vertices 3 and 4 retain these high degree centrality values across much more of the filtration than vertex 1, suggesting they are more important to the network, despite having the same centrality metric on the unfiltered network (column 1).

Now, let's add some code to calculate degree centrality across Graph 2's filtration:

```
#calculate degree centrality for Graph 2's filtration sequence
d2<-degree(g2)
d22<-degree(g22)
d24<-degree(g24)
d26<-degree(g26)
d28<-degree(g28)

#create a dataset tracking degree centrality across the filtration
g2deg<-cbind(d2,d22,d24,d26,d28)
```

This code calculates degree centrality across the filtration of Graph 2, yielding a table similar to that obtained by Graph 1's filtration and

centrality calculation. Table 4-2 summarizes the findings from the Graph 2 filtration and centrality calculation.

Table 4-2: Degree Centrality Across Graph 2

Column1	d2	d22	d24	d26	d28
V1	4	3	3	3	3
V2	4	4	3	2	1
V3	4	4	3	2	1
V4	5	4	3	0	0
V5	3	3	1	1	0
V6	4	4	3	2	1

As Table 4-2 shows, there are relatively high degree centrality measures in the unfiltered Graph 2; however, the pattern changes by vertex after the filtration begins. Some vertices, like vertex 1, retain a high degree centrality throughout the filtration. Others, such as vertex 4, retain a high degree centrality and then drop to 0. Others still, like vertex 6, show a slow degradation of degree centrality over the full filtration. This may be informative in a study of social ties within a subgroup of interest. A high degree of informal social ties, represented by a high centrality degree, has been linked to positive educational attainment, career achievement, and resilience to life adversity in young adults.

Degree centrality is only one example of metrics that we can calculate across a filtration; we can also calculate other local metrics such as betweenness centrality or triadic closure. In addition, we can calculate global metrics, such as the spectral radius or the Euler characteristic, across a filtration. Let's add the following to Listing 4-1 to calculate the diameter of each filtration of Graph 1:

```
#calculate graph diameter of Graph 1's filtration
di1<-diameter(g1)
di12<-diameter(g12)
di14<-diameter(g14)
di16<-diameter(g16)
di18<-diameter(g18)
```

The sequence of diameters calculated across the filtration of Graph 1 by this code is 2.1, 2.9, 2.9, 1.6, and 0.9. Let's calculate the diameters for Graph 2's filtration:

```
#calculate graph diameter of Graph 2's filtration
di2<-diameter(g2)
di22<-diameter(g22)
di24<-diameter(g24)
di26<-diameter(g26)
di28<-diameter(g28)
```

The sequence of diameters calculated across the filtration of Graph 2 by this code is 0.9, 1.2, 1.6, 2.4, and 1.7. This is different than Graph 1's diameter sequence, suggesting that the diameter is generally smaller until later in the filtration sequence. This metric's filtration might be useful in assessing a community's overall level and depth of informal social ties, a measure of community resources available to residents in need. Figure 4-2 shows the diameter plots across both filtrations to compare the two networks.

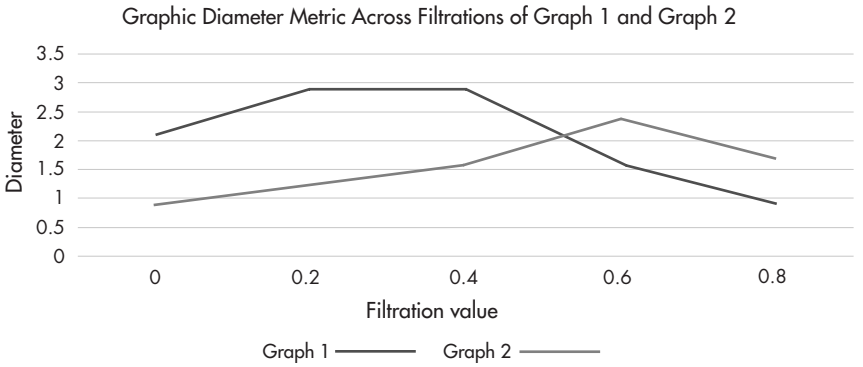

Figure 4-2: A plot of graph diameter metrics across filtrations of Graph 1 and Graph 2

As we can see in Figure 4-2, Graph 1 has a larger graph diameter than Graph 2 early in the filtration, but this relationship switches after a filtration value of 0.4. This suggests that there is greater eccentricity in Graph 1 early in the filtrations but greater eccentricity in Graph 2 later in the filtration. Remember that eccentricity is the maximum distance from one point to another in the network.

Graph filtration tracking as we've plotted in Figure 4-2 can be helpful in distinguishing similar graphs with different connectivity patterns or weights. Dynamic networks, in which weights can change over time, could be a use case of graph filtrations. In addition, they are quite useful in comparison among networks with the same vertices but potentially different weights (such as patient groups in brain imaging studies); in fact, brain imaging studies are one of the applications for which graph filtration was developed. Higher eccentricity values suggest longer pathways to relay neural signals; stronger edge weights represent stronger connections between two areas of the brain. Strong edges with low eccentricity suggest a functional module activated in a particular task given to the patient groups on which imaging was performed.

Although graph filtration is a relatively new concept, it has mainly been confined to biological network data, including networks based on brain imaging studies. However, the graph filtration method is widely applicable to weighted network data, and its tool set lends itself to further development in other fields. If you want to explore this topic in more depth, look through the references at the end of this book and play around with graph

filtrations on their own data. For now, let's turn our attention to a topological view of graphs, which allows us to extend the relationships captured in graphs to other types of interactions between people or things.

Simplicial Complexes

Graphs can be considered topological objects that have defined global properties we can leverage in our analyses, and it's possible to turn a graph into a higher-dimensional version of a graph, called a *simplicial complex*, by considering three-way, four-way, and *n*-way interactions by individuals and vertices in the graph. Let's consider three colleagues who often collaborate on academic papers but have never published with all three names on a paper. We'll create a simple graph for the three colleagues, shown in Figure 4-3.

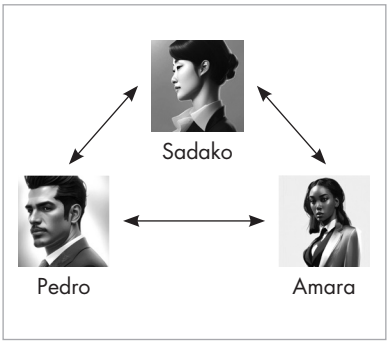

Figure 4-3: A simplicial complex showing two-way interactions among three colleagues

Now let's imagine a paper where all three colleagues participate and have their names on the paper. This is a three-way interaction, rather than three two-way interactions, and we'd end up with a filled-in triangle rather than three sets of two-way arrows, as shown in Figure 4-4.

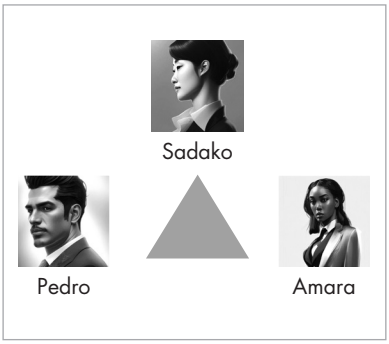

Figure 4-4: A simplicial complex showing three-way interactions among three colleagues

Figure 4-4 uses a triangle to represent a three-way connection among colleagues, similar to how the arrows between two colleagues represented two-way connections. This can be generalized to tetrahedra for four-way interactions and more exotic shapes to represent higher n-way interactions. There's no limit as to how high of a number n can be, but computational issues will come into play at some point as we work our way up to n-way interactions in a simplicial complex. Analyses involving email chains, co-authors on papers, or conference calls are common applications that extend social network analysis and graphs into the analysis of simplicial complexes. Depending on the size of the network and the size of the n-way interactions, simplicial complex representations of individuals and mutual interactions can become very complicated across values of n. Analyzing these structures can involve a lot of computing power and tools that extend network metrics. However, because graphs are topological objects, many theorems and tools of topology can be successfully applied to them without transformations or other hassles. This, in turn, allows for other areas of math, including partial differential equations and probability theory, to be applied and developed on graphs.

Just as we could filter a weighted graph, we also can filter simplicial complexes. The filtration process for simplicial complexes varies depending on how the simplicial complex is built. In most topological data analysis algorithms, we start with a point cloud of data within a space where a distance metric can be defined. Points are included in a simplicial complex if they share either mutual n-way overlapping sets with each other (*Čech complex*) or pairwise overlapping sets (*Vietoris–Rips complex*). By sequentially increasing or decreasing the value of the distance metric, we obtain a filtration of simplicial complexes. In practice, the Vietoris–Rips complex is easier to compute and underlies many common topological data analysis packages. This leads us to a very new and emerging part of network analytics: extensions of network tools to simplicial complexes.

Many of the tools introduced in the previous chapters have simplicial complex analogs, including eccentricity, shortest path algorithms, centrality metrics (Katz centrality, eigenvector centrality, closeness centrality, and so on), triadic closure, and many more. Typically, simplicial complexes of network data are built by computing maximal cliques within the network (though it's possible to define a distance metric and apply the process defined in the prior paragraph to build simplicial complexes from network data as well). *Maximal cliques* of a network include the highest n-way mutual edges among groups of vertices. These maximal cliques correspond to an (n-1)-simplicial complex. The *flag complex* of the graph involves building the graph's simplicial complex by computing the graph's maximal cliques. From this complex, it's possible to define quantities at each simplicial complex level, which can be combined into a total metric across levels. This means we can glean more information about the overall structure of the network and its components at various levels of a simplicial complex.

Let's return to Farrelly's social network introduced in prior chapters and look at an extension of degree centrality, dubbed *topological dimension*. We can define topological dimension as a weighted degree centrality,

weighting each vertex by the dimension of the cliques in which it resides, which involves summing across a vertex's cliques of different dimensions. For instance, a vertex in a maximal two-clique and a maximal three-clique within the network would have a topological dimension of 5. A vertex in a maximal five-clique and no other cliques would also have a topological dimension of 5. However, the former vertex might have a degree of 3, connecting to one other vertex in the two-clique and two other vertices in the three-clique; the latter would have a degree of 4, connecting to the four other vertices in the five-clique.

In Listing 4-2, we have a script that calculates the maximal cliques and the topological dimension of vertices within Farrelly's social network.

```
#load the author's network
g_social<-read.csv("SocialNetwork.csv")

#create the graph
library(igraph)
g1<-graph_from_adjacency_matrix(g_social,mode="undirected",weighted=F)

#compute the maximal cliques in the author's network data
cl<-maximal.cliques(g1)

#create array
cl<-as.array(cl)

#get clique size from maximal clique array
d<-dim(cl)
l<-rep(NA,d)
for (i in 1:d){
l[i]<-length(as.vector(cl[[i]]))}

#create matrix of vertices in maximal cliques
av<-matrix(rep(NA,d*20),20)
for (i in 1:20){
for (j in 1:d){
av[i,j]<-i%in%cl[[j]]
}}

#convert to binary indicators
avind<-ifelse(av==TRUE,1,0)

#multiply out to calculate each vertex's topological dimension
topmat<-t(avind)*l
topdim<-colSums(topmat)
```

Listing 4-2: A script that calculates topological dimension across vertices in Farrelly's social network

This script results in a topological dimension calculation based on the flag complex of the graph. It first calculates the flag complex from the maximal cliques; it then stores the information of each clique, such that we can cycle through each clique to see which vertices belong to each clique. Converting this information to a binary indicator matrix allows us

to multiply the dimension of the clique and the indicator matrix, resulting in a vector containing the topological dimension of each vertex. Table 4-3 shows the topological dimension and degree of each vertex in the author's network dataset.

Table 4-3: Topological Dimension and Degree Summary for Vertices in Farrelly's Social Network

Vertex	Degree	Topological dimension
1	2	3
2	1	2
3	5	11
4	2	3
5	4	7
6	3	6
7	8	18
8	3	4
9	3	4
10	3	6
11	3	5
12	1	2
13	4	8
14	4	7
15	4	8
16	2	4
17	2	4
18	3	6
19	2	4
20	1	2

Table 4-3 shows a distinct difference between degree, which includes only the vertices and edges of the author's network in its calculation, and the topological dimension, which includes higher-order interactions. For instance, vertices 9 and 10 both have a degree of 3; however, their topological dimensions differ, with vertex 9 having a score of 4 and vertex 10 having a score of 6. The importance of vertex 10 to the overall network structure is larger than the importance of vertex 9 to the overall network structure. Without considering higher-order interactions within the network, we would not be able to distinguish between the two vertices with respect to this metric.

For weighted networks, it's possible to combine these simplicial-complex-based metrics with graph filtration, yielding a sequence of metrics over the

filtration based on the simplicial complex of the network. You'll see this when we discuss a tool called *persistent homology* in the next section of this chapter. You could do the same with the Euler characteristic or the topological dimension or a yet-to-be-developed simplicial complex extension of network metrics.

Simplicial complex extensions of network metrics are a very new area of study within network science, and few packages or open source functions exist to calculate the simplicial analogs of network metrics. However, it is hoped that this example and some of the papers on this topic will spark the addition of simplex-based metric within network science packages. Perhaps you will take up the challenge and contribute functions to the igraph package or other open source network science tools.

The next tools we look at will involve a bit more topology than we've encountered so far, so first let's explore another topological concept that's useful in graph analytics and in understanding simplicial complexes.

Introduction to Homology

The basic topological premise of our next set of tools involves counting different dimensions of holes in an object or dataset. Consider a piece of paper with a hole in the middle of it or a basketball with a sphere of air inside it. These are holes of different dimensions, and each hole separates connected pieces of an object from other pieces of itself. When these holes exist in manifolds or functions, we can systematically study them and classify objects or spaces based on the number and dimension of these holes.

Homology is the counting of varying-dimensional holes (connected components, circles, spheres, voids, and so on) within a given object or space, usually to classify that object or space. For low-dimensional spaces, this is fairly straightforward; you can actually build a physical model of the space and count the holes. However, there are also variants of homology that allow topologists to distinguish between different types of objects and spaces that may be higher dimensional or strangely shaped without requiring a physical model.

Numbers corresponding to holes in each dimension create a handy collection of values, called *Betti numbers*, that organize the number and type of hole within a given object or space such that each object can be classified and studied alongside other objects whose numbers match. If you're familiar with algebraic topology, this is a standard procedure for the classification of abstract mathematical structures. Commonly, these numbers are stored in a vector. It's a bit abstract, but we'll go through some simple examples.

Examples of Betti Numbers

Many sports involve using a ball, but not all balls are the same, topologically speaking. Basketballs and baseballs are both round balls in

three-dimensional space. Basketballs are usually bigger than baseballs, but if there were a child's toy basketball of the same size as a baseball, one might look at them and think they are quite similar.

Figure 4-5: An example baseball and basketball, which look similar but are topologically distinct

Topologically, though, they are quite distinct. These two balls differ in second Betti numbers, which count three-dimensional voids in an object. A vector of Betti numbers is an infinite sequence of numbers representing the number of holes in each dimension, starting with connected components on the zeroth number position and moving to circles (first number position), voids (second number position), and higher-dimensional voids (starting from the third position and going to infinite position). In practice, most datasets don't have many holes past the first Betti number, so we can fill the rest of the vector with zeros. The hollow basketball has a hole past the first Betti number because it contains a void, giving a vector of Betti numbers (1, 0, 1, 0, . . .), while the solid baseball has no holes of any dimension, corresponding to a Betti number vector of (1, 0, 0, 0, . . .).

Some objects have more than one hole in a given dimension. For instance, imagine gluing a second basketball to the outer surface of the basketball in Figure 4-5. This object would obviously have another void, yielding a Betti number vector of (1, 0, 2, 0, . . .). A donut, or *torus*, has a vector of (1, 2, 1, 0, . . .), as it has two open circles defining the ends of the tube, which form a void when connected at the ends. Figure 4-6 shows the classical construction of a torus from a sheet of paper.

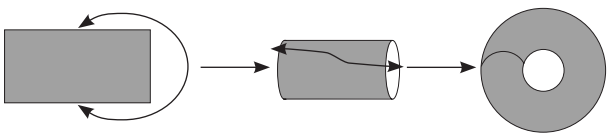

Figure 4-6: The construction of a torus from a sheet of paper connected at the edges

It's fairly easy to classify objects and spaces that can be easily visualized in three dimensions. However, many datasets used in the industry involve more than three dimensions, and comparisons and classifications of these objects require algorithms that can discern the Betti numbers associated with those objects; among these are genomics datasets (which can involve million-dimensional spaces), video sequences, and multivariate time series.

The Euler Characteristic

One of the topology-based metrics shows up both in the analysis of networks and in their higher-dimensional simplicial complex cousins, and it ties back to the notion of curvature introduced in prior chapters. The *Euler characteristic*, often given the notation of χ, provides a single number to summarize a topological space and is a topological invariant, meaning that the topological quantity being calculated does not change as the space is continuously deformed (stretched, twisted, or otherwise manipulated without tearing the space). The Euler characteristic can be defined using Betti numbers; technically, computing the Euler characteristic this way involves an alternating sum of Betti numbers (zeroth Betti number – first Betti number + second Betti number – third Betti number + fourth Betti number . . . up until the highest Betti number that exists).

The Euler characteristic can also be defined through the dimensions of the simplicial complex (number of vertices – number of edges + number of triangles – number of mutual 4-way interactions + . . .). However, vertices included in an edge aren't counted in the number of vertices. A triangle that makes up part of a mutual four-way interaction won't be counted either.

However, there is an easy way to obtain the largest pieces of a network or its higher-dimensional simplicial complex using an igraph function related to maximal cliques (as mentioned earlier). Maximal k – cliques denote and count the k – 1 simplices of the full simplicial complex derived from the network. They're a convenient way to build the full simplicial complex and keep track of the pieces involved at each n-way interaction. Let's add to the script in Listing 4-2 to count the maximal cliques in the author's network:

```
#create a table counting the number of k+1 simplices in the simplicial complex
summ<-as.numeric(summary(cl)[,1])
jjj<-table(summ)
```

This code creates a table summarizing the maximal cliques in the network that we previously computed. The result should yield 11 two-cliques (one-simplices, or edges), 6 three-cliques (two-simplices, or triangles), and 1 four-clique (four-simplices, or a mutual four-way interaction). We can plug these values into the Euler characteristic formula:

$$\chi = 0 \text{ vertices} - 11 \text{ edges} + 6 \text{ triangles} - 1 \text{ tetrahedron}$$

This gives a χ of –6. Recent studies have shown that most real-world networks have negative Euler characteristics. There's a very interesting reason that network data tends toward negative Euler characteristics related to the curvature of the network. Negative curvature in graphs is associated with the robustness of the network; biological networks with highly negative curvature can often withstand loss of function within

parts of the network without adverse effects on the organism. The *Gauss–Bonnet theorem* relates the Euler characteristic, defined through homology, and the curvature of the object, including the manifold's curvature and the curvature of the manifold's boundary. There have been some recent attempts to link network analytics tools such as homology and Forman–Ricci curvature for a deeper study into network properties. This is a deep result in a branch of mathematics called *differential geometry* that connects an object's local geometry to its global topology, and it's a newer area of study in network science. Now that we know network topology and geometry are related to each other, let's look at a topological tool called *persistent homology*.

Persistent Homology

One of the most common topology-based algorithms used in data analysis today is persistent homology, which has been applied in genomics, healthcare, economics, energy, psychometrics, and many other fields. In essence, the idea of the persistent homology algorithm is to build a point cloud from the data, filter it into a series of simplicial complexes based on different thresholds of the data (akin to an MRI), and track topological features, such as holes or voids, appearing and disappearing in each slice. For instance, consider the three slices of cheese in Figure 4-7, each containing holes in the shape of circles; these circles affect the first Betti numbers of the datasets.

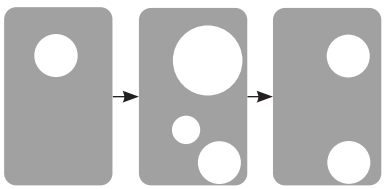

Figure 4-7: Three slices of a cheese block containing holes in different places

In Figure 4-7 one hole appears in all three slices, another appears in only the middle slice, and one appears in two slices. Holes and voids can be of different sizes in real data, and as we move across slices, holes might grow or shrink in diameter. Persistent homology algorithms have thresholds for both the lifetime of a feature and the minimum size considered for measuring a hole. In our example, we have features that are likely noise (either too small of radius or only appearing in one slice of our cheese) and features that are likely real features in the dataset (such as the void appearing in all three slices). Let's unpack this intuition.

Say we want to compare two datasets to see whether they are collected from the same distribution or shape. This is common when matching image

data. While image data rarely comes with cheese holes, circles come up in image data quite frequently in the form of eyes.

Technically speaking, by varying the distances used to build the simplicial complex from the point cloud data (or filtering), you can track various Betti numbers through the filtration and assign each hole in the data an importance score, with important features lasting over longer filtration distances (longer *persistence*, in the parlance of persistent homology). In Figure 4-7, the hole that appears in all three slices would be considered the most important feature, and the hole that appears in only the second slice might be a result of noise in the data. These features can then be plotted on a *barcode* or *persistence diagram* that tracks these features' lifetimes (distance scale over which they exist in the filtration). We'll explore barcodes and persistence diagrams in the following example analysis.

In practice, datasets are usually examined only for low-dimensional holes and features due to computational issues, and the zeroth (connected components) and first (circles) Betti numbers are used most commonly unless you are explicitly computing high-dimensional shape data. The example in Figure 4-7 is connected in all three slices, so it has a zeroth Betti number of 1 across all slices. However, circles appear and disappear through the filtration, giving a barcode that looks like Figure 4-8.

Figure 4-8: A diagram plotting the persistence of features (holes) captured in the box of Figure 4-6

The barcode shows the time at which features appear and disappear. For instance, in Figure 4-8, we can see a feature that appears at time 2 and disappears at time 3 (our bottom cheese hole in Figure 4-7). The sequence of connected components across the data slices has a curious relationship with another machine learning method, single-linkage hierarchical clustering, in which clusters at each height level correspond to the connected components at that particular slice. When both techniques use the same distance metric, the results are actually identical; however, the persistent homology approach will give more information than single-linkage hierarchical clustering's dendrogram regarding the structure of the data. This means that machine learning practitioners can choose the technique that fits the problem best, as these two options come with their own plots and statistical tests. For instance, with a nontechnical audience, single-linkage hierarchical clustering might be preferable, as dendrograms and heatmaps are more familiar to biologists or social scientists.

Comparison of Networks with Persistent Homology

Within the realm of network analytics, persistent homology can be a useful way to compare network structures to see if different networks have the same underlying geometry. Let's explore this further with an application to simulated networks. In neuroscience, it's common to translate fMRI or PET data into a network structure, where different regions of the brain are translated to vertices and connected to other regions of the brain based on activity patterns (sequential activation of an area, for instance, or co-activation of multiple regions during one task). Often, outcomes of interest involve comparing groups of patients, either healthy patients against a group of patients with a particular neurological or psychological disorder or two disease groups, to understand differences in the brain activation patterns across disorders.

We'll explore the use of persistent homology in the comparison of two such networks. Because fMRI data isn't readily available as open source, we'll simulate networks in igraph that are approximately the size of brain imaging networks; this will demonstrate how this methodology would be applied to imaging data that has been transformed to network data.

The igraph package allows you to simulate many types of network data, including Erdös–Renyi graphs, scale-free graphs, and Watts–Strogatz graphs. We'll create each of these types of graphs using the script in Listing 4-3.

```
#simulate three graphs using the igraph package for further comparison
library(igraph)

#create an Erdos-Renyi graph
g1<-erdos.renyi.game(30,0.3)

#create a scale-free graph
g2<-sample_pa(30,power=2.5,directed=F)

#create Watts-Strogatz graph
g3<-sample_smallworld(2,5,3,0.3)

#plot the three graphs created
plot(g1,main="Erdos-Renyi Graph")
plot(g2,main="Scale-Free Graph")
plot(g3,main="Watts-Strogatz Graph")
```

Listing 4-3: A script that simulates three different types of network structures for statistical comparison

Listing 4-3 creates three different types of networks that can later be compared via persistent homology; it also visualizes the networks, which should yield something similar (but probably not identical) to Figure 4-9.

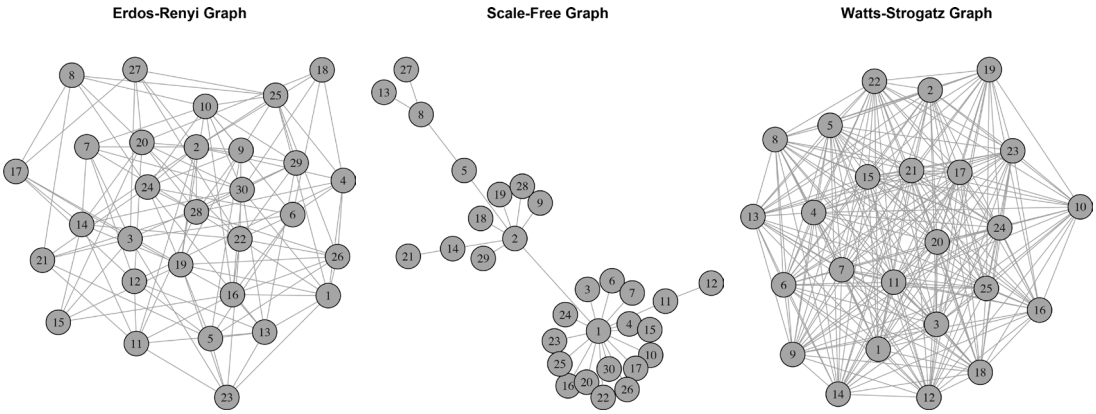

Figure 4-9: Plots of the three simulated network types

Figure 4-9 shows very different types of graphs. The scale-free graph in the middle includes a hub with many vertices connected to the hub but not to other vertices. The Erdös–Renyi graph on the left and the Watts–Strogatz graph on the right have many more interconnections, but the Watts–Strogatz model seems to have more structure connecting vertices into cliques, rather than randomly connecting vertices.

Let's apply persistent homology to these networks and compare the distance between persistence diagrams among these networks by adding the following to Listing 4-3; again, your results may vary given the simulation of each network type:

```
#load TDA package
library(TDAstats)

#get adjacency matrices
m1<-as.matrix(get.adjacency(g1))
m2<-as.matrix(get.adjacency(g2))
m3<-as.matrix(get.adjacency(g3))

#compute persistent homology
d1<-calculate_homology(m1, dim = 2, format = "cloud")
d2<-calculate_homology(m2, dim = 2, format = "cloud")
d3<-calculate_homology(m3, dim = 2, format = "cloud")

#plot persistence diagrams
plot_persist(d1)
plot_persist(d2)
plot_persist(d3)

#compute distances among graphs
w1<-phom.dist(d1,d2,limit.num=0)
w2<-phom.dist(d1,d3,limit.num=0)
w3<-phom.dist(d2,d3,limit.num=0)
```

This addition derives an adjacency matrix from each of the simulated graphs and computes a persistence diagram from this adjacency matrix, which is then compared through the distances between the zeroth homology groups. This script should produce three persistence diagrams that look like Figure 4-10 (note they won't be identical, as each run will produce something slightly different).

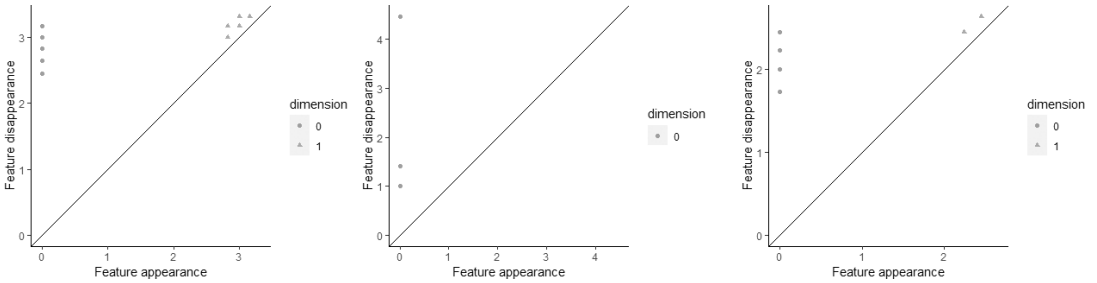

Figure 4-10: Persistence diagram plot for the three simulated network types (from left to right: Erdös–Renyi, scale-free, and Watts–Strogatz)

Figure 4-10 shows varying topological features found in each of the network types. The Watts–Strogatz network and Erdös–Renyi graphs both produce many large zeroth homology features (the dots), while the scale-free graph has a variety of zeroth homology feature sizes. The scale-free graph does not have higher-order homology features, while the other two graphs have first homology features (the triangles), albeit very near the diagonal line (suggesting that they may be noise). A point directly on the diagonal line is a feature that is in only one slice of the data; the further from the diagonal line a point lies, the longer it has existed in the data. With respect to our three simulated networks, it's hard to tell if the scale-free and Watts–Strogatz graphs differ significantly from the Erdös–Renyi graph just by looking at the persistence diagrams.

We can add to our script to derive a null distribution for the Erdös–Renyi persistence diagram and use a special distance metric, Wasserstein distance, to statistically test the structural differences between the Erdös–Renyi persistence diagram and the scale-free and Watts–Strogatz persistence diagrams:

```
#get Wasserstein distance between random graphs with the same structure
ww<-rep(NA,100)

for (i in 1:100){
g1<-erdos.renyi.game(30,0.3)
g2<-erdos.renyi.game(30,0.3)
m1<-as.matrix(get.adjacency(g1))
m2<-as.matrix(get.adjacency(g2))
```

```
d1<-calculate_homology(m1, dim = 2, format = "cloud")
d2<-calculate_homology(m2, dim = 2, format = "cloud")
ww[i]<- phom.dist(d1,d2,limit.num=0)}

#compute 95% confidence intervals from the simulated null distribution
quantile(ww,c(0.025,0.975))
```

This script creates a null distribution of Erdös–Renyi persistence diagrams from the same distribution that the original persistence diagram was constructed from; your results may vary, given the random component to the simulation piece. Quantiles of our null distribution give a confidence interval of (0.91, 8.36), which includes quite a bit smaller distances than the distances computed between the persistence diagrams of the Erdös–Renyi graph and the Watts–Strogatz graph (23.59) and between the persistence diagrams of the Erdös–Renyi graph and the scale-free graph (39.78). Thus, we can conclude that the structures of the Watts–Strogatz graph and the scale-free graph are not random. There is a significant structural component to each of these graphs.

This type of simulation can be very useful in testing differences between persistence diagrams of brain networks derived from fMRI and PET imaging studies, and it's easy to implement in R. This methodology can also be applied to other networks with a hypothesized underlying structure, such as social networks or power grids. Many other types of network analysis tools can also be used to compare graph structures, such as local and global metrics (including graph radius and diameter, degree distributions, clustering graph coefficients, and so on), and many of these comparisons haven't been explored much yet.

Summary

In this chapter, we filtered weighted networks to understand how network metrics change as edges are removed based on their weights. Then, we built simplicial complexes from network data to leverage several topological tools, including an extension of the degree metric, the Euler characteristic, and a filtration-based algorithm called persistent homology that can be used to compare networks. In the next chapter, we'll transition from network science to distance geometry as we explore how different measurement choices impact supervised and unsupervised learning algorithms.