Manual of the topologyR Package: Topological Structures Analysis

José Mauricio Gómez Julián

2024-12-30

Contents

1. Theoretical Foundations **2**
1.1. Introduction to Topological Spaces . . . 2
1.1.1. Formal Definition . . . 2
1.1.2. Fundamental Properties . . . 2
1.2. Bases and Subbases . . . 2
1.2.1. Definition and Properties of Bases . . . 2
1.2.2. Subbases and their Relationship with Bases . . . 3
1.3. Connectivity in Topological Spaces . . . 3
1.3.1. Topological Definition of Connectivity . . . 3
1.3.2. Connectivity in Graphs and its Relation to Topological Connectivity . . . 4
1.4. Intuitive Explanation of Theoretical Foundations . . . 4

2. Algorithm Implementation **5**
2.1. Base Algorithm . . . 5
2.2. Implementation Considerations . . . 9
2.3. Optimizations and Heuristics . . . 9

3. Package Functions **13**
3.1. is_topology_connected() . . . 13
3.2. is_topology_connected2() . . . 13
3.3. is_topology_connected_manual() . . . 14
3.4. analyze_topology_factors() . . . 14
3.5. calculate_thresholds() . . . 15
3.6. visualize_topology_thresholds() . . . 16

4. Applications **19**
4.1. Economic Time Series Analysis . . . 19
4.1.1. Data Preparation and Distribution Analysis . . . 19
4.1.2. Basic Topology Analysis . . . 24
4.1.3. Advanced Analysis with Thresholds . . . 25
4.1.4. Performance Optimization Guidelines . . . 29
4.1.5. Interpretation Framework . . . 30
4.2. Comparative Analysis Examples . . . 31
4.3. Some Recommendations for Your Dataset . . . 32
4.3.1. Data Preparation . . . 32
4.3.2. Error Handling: . . . 33
4.3.3. Optimization for Large Datasets . . . 33

5. Limitations and Considerations **34**
5.1. Computational Limitations . . . 34
5.2. Approximations and Validity . . . 35

1. Theoretical Foundations

1.1. Introduction to Topological Spaces

1.1.1. Formal Definition

To understand the algorithm implementation, fundamental topological concepts must first be comprehended, specifically those of base, subbase, and their relationship with an object's topology.

Let X be an arbitrary set. A *topology in* X is any system τ of subsets G of X that verifies two conditions (Kolmogórov & Fomin, 1978, p. 90):

1. The set X itself and the empty set $\emptyset$ belong to τ.
2. The union $\bigcup_\alpha G_\alpha$ of any number (finite or infinite) and the intersection $\bigcap_k G_k$ of a finite number of sets in τ belong to τ. Another way to define it, following Kelley (1955, p. 50), is that the intersection of any two members of τ is a member of τ (τ is closed under intersections) and that the union of the members of any family of τ is part of X (the topology space of τ is closed under arbitrary unions).

Thus, the set X is called the *topology space of* τ, τ is called the *topology of* X, and the pair (X, τ) is called a **topological space**, that is, $T = (X, \tau)$. The sets belonging to the system of sets τ are called *open* (Kolmogórov & Fomin, 1978, p. 90).

As Kolmogórov & Fomin (1978, p. 90) point out, a metric space consists of a set of points and a metric introduced in that set; similarly, a topological space consists of a set of points and a topology introduced in it. Consequently, defining a topological space means defining a set X and a topology τ on it, that is, indicating which subsets are considered open in X (Kolmogórov & Fomin, 1978, p. 90).

It is clear that different topologies can be introduced in the same set X, thus transforming it into different topological spaces (Kolmogórov & Fomin, p. 90). However, the topological space, that is, the pair (X, τ), will be denoted by T, and the elements of the topological space will be called "points".

1.1.2. Fundamental Properties

The sets $T \setminus G$, that is, the elements of T that do not belong to G, which are complementary to the open sets, are called *closed* sets of the topological space T (Kolmogórov & Fomin, 1978, p. 90).

The principle of duality consists of the fact that from any theorem concerning a system of subsets of a fixed set S, another dual theorem can be automatically deduced by replacing the original sets with their complements, the sum of sets with their intersection, and the intersection with the sum (Kolmogórov & Fomin, 1978, p. 16). By virtue of the duality relations, from axioms 1 (the complement of the sum is equal to the intersection of the complements) and 2 (the complement of the intersection is equal to the sum of the complements), it follows that (Kolmogórov & Fomin, 1978, p. 90):

1. The empty set $\emptyset$ and the entire space T are closed.
2. The intersection of any number (finite or infinite) and the union of a finite number of closed sets are closed.

In every topological space, the concepts of neighborhood, adherence points, set adherence, etc., are naturally introduced from these definitions.

1.2. Bases and Subbases

1.2.1. Definition and Properties of Bases

As Kolmogórov & Fomin (1978, p. 91) point out, a *neighborhood* of the point $x \in T$ is any open set $G \subset T$ that contains the point x; a point $x \in T$ is called an *adherence point* of the set $M \subset T$, when every neighborhood

of point x contains at least one point of M; a point x is called an *accumulation point* of the set M, when every neighborhood of the point x contains an infinite number of points of M. The totality of adherence points of the set M is called the *adherence* of the set M and is denoted by the symbol $[M]$ (Kolmogórov & Fomin, 1978, p. 91).

As Kolmogórov & Fomin (1978, p. 93) note, a collection ζ of open subsets is called a *base* of the topological space T, when every open subset of T can be represented as a certain number of sets from ζ.

> For example, the collection of all open balls (of all possible radii and centers) constitutes a base in a metric space. In particular, the system of all intervals is a base on the line. From the above, it follows that the topology τ of the space T is defined if a base ζ is indicated in this space. (Kolmogórov & Fomin, 1978, p. 93)

As Kolmogórov & Fomin (1978, p. 93) point out, for this method of introducing topology to have practical value, it is necessary to indicate those conditions that a system ζ of subsets of the given set T must satisfy so that the collection of all possible sums of sets from ζ can be considered as the collection of open sets in T (that is, so that these sums verify axioms 1° and 2° of topological space). Such conditions are given by the following theorem:

THEOREM Suppose that in a set T, a system ζ of subsets G_α has been chosen that verifies the following conditions:

a. Every point $x \in T$ is contained in at least one subset $G_\alpha \in \zeta$

b. If $x \in G_\alpha$ and $x \in G_\beta$, there exists a $G_\gamma \in \zeta$ such that $x \in G'_\gamma \in G_\alpha \bigcup G_\beta$, where G'_γ is the complement set of G_γ, that is, $G'_\gamma = T \setminus G_\gamma$.

Thus, if we declare open in T the empty set and all sets that can be represented as the sum of certain $G_\alpha \in \zeta$, the set T will result in a topological space (that is, these sums will verify axioms 1° and 2°) and the system ζ will be a base of it (Kolmogórov & Fomin, 1978, p. 93).

1.2.2. Subbases and their Relationship with Bases

To prove whether a given collection of open sets is a base or not, the following criterion is often useful (Kolmogórov & Fomin, 1978, p. 94):

THEOREM

For a system G_α of open sets to be a base of the topological space T, it is necessary and sufficient that for every open set G and every point $x \in G$, there exists a set G_α from this system such that $x \in G_\alpha \in G$.

Finally, Kelley (1955, p. 61) notes that a family S of sets is a **sub-base of a topology** τ if the family of all finite intersections of members of S is a base of τ (or equivalently: every member of τ is a union of finite intersections of members of S). A more intuitive way to see it is that a subbase is a collection S of subsets of a topological space that is contained in a base of the topology and can be completed to form a base by adding all finite intersections of subsets of the set S.

1.3. Connectivity in Topological Spaces

1.3.1. Topological Definition of Connectivity

Kelley (1955, p. 67) notes that a topological space $T = (X, \tau)$ is connected if X is not the union of two separated non-empty subsets. In other words, it is a subset of a topological space that cannot be described as a disjoint union of two non-empty open sets of the topological space in question.

The above means that it cannot be simultaneously true that, on one hand, if we separate the topological space into two sets, that is, the endpoints of each new set (resulting from the separation) are not included in these two sets, upon reuniting them the result is equivalent to the original set and, on the other hand, that this intersection is empty (that they have no elements in common).

This does not occur simultaneously because if they are separated into two subsets, it is possible to achieve that upon intersecting them (reuniting) the result is an empty set, but only at the cost of omitting some point from the original sets; on the other hand, it is possible to return to the original set, but only at the cost of not omitting any of the original points, in which case their intersection would not be an empty set.

The above can also be formally expressed by saying that a connected set is a subset $C \subseteq X$ of a topological space (X, τ) that cannot be described as a disjoint union of two non-empty open sets of the topology, where T is the collection of open sets of the topological space; this subset, as its own definition reveals, may be equivalent to the complete topological space. In other words, it is formed by a single piece and is not divisible because all its parts are connected in some way, so analyzing parts in isolation will affect the global properties of the set.

By contrast, in a non-connected or disconnected set, the situation is different. A non-connected set can be separated into two or more disjoint open subsets, each of which is called a connected component.

This means that in disconnected sets, it is possible to analyze a part (i.e., a connected component) of the set independently, and in many cases, this will not affect the global properties of the set. However, there are important nuances.

First, local properties can be completely analyzed in a connected component without affecting the others. Second, some global properties will be maintained when analyzing a single component, while others will require considering all components; for example, the compactness of a component does not imply the compactness of the total set.

Some topological (i.e., essential) properties that can be analyzed locally (by components) are the connectivity of each component, the compactness of each component, or other more general local topological properties (for example, being locally Euclidean). In contrast, global properties that will require considering all components, even when the set is disconnected, include the total number of connected components, the compactness of the total set (which requires all components to be compact and that there be a finite number of them), different metric or measure properties (to see if they are generalizable to the entire space), among others. Thus, although it is possible to analyze components separately, it is crucial to remember that some global properties can only be understood by considering all components together.

The main difference between connected and non-connected sets in terms of local properties is that in a connected set, local properties can "propagate" or have global implications in ways that do not occur in non-connected sets. For example, consider a continuous function f on a closed interval $[a, b]$ predefined as connected. It is possible to analyze local properties such as differentiability at each point; however, the Intermediate Value Theorem, which is a global property, is derived from local continuity and the connectivity of the interval: consequently, what is not possible in a connected set is to completely isolate a part of the set from the rest, in topological terms, since there will always be some form of "connection" between the parts.

1.3.2. Connectivity in Graphs and its Relation to Topological Connectivity

Once the topology (complete or approximated by optimization) is revealed, its connectivity can be determined by constructing a directed graph or an undirected graph.

A graph is a set of objects called vertices or nodes connected by links called edges or arcs, which allow representing binary relations between elements of a set (Trudeau, 1993, pp. 19-20).

An undirected graph is justified when it is valid to assume that the connection relationship between the topology elements is symmetric. In other words, if element A is connected to element B, then B is also connected to A.

1.4. Intuitive Explanation of Theoretical Foundations

Imagine we have a set X representing the Cartesian plane. Within this plane, we draw a circle with its corresponding circumference. The circumference is the line that delimits the circle, while the circle is the area enclosed by the circumference.

Now, let's introduce a topology τ in this set X. The topology τ is a collection of subsets of X that meets certain conditions; the Cartesian plane would be the underlying space from which the topology is constructed, while the plane plus the topology would be the topological space. In our example, we can consider that the open sets in this topology are the circle (without including the circumference) and the area outside the circle (excluding the circumference). These open sets satisfy the mentioned conditions: the set X (entire plane) and the empty set belong to τ, and arbitrary unions and finite intersections of open sets are also open.

On the other hand, the closed sets in this topology would be the circumference and the complete set X (including the circle and its exterior). These closed sets satisfy the dual conditions: arbitrary intersections and finite unions of closed sets are also closed.

Now, imagine a point x inside the circle. A neighborhood of x would be any open set that contains x, that is, any area within the circle that includes x. A point y on the circumference would be an adherence point of the circle, since any neighborhood of y (any open area containing y) will always include at least one point of the circle. The adherence of the circle would be the circle along with its circumference, as it includes all adherence points.

A base ζ of the topology τ would be a collection of open sets such that any open set in τ can be expressed as a union of elements of ζ. In our example, a base could be the collection of all open disks within the circle.

Finally, a sub-base S of the topology τ would be a collection of sets such that all finite intersections of elements of S form a base of τ. In our example, a sub-base could be the collection of all open regions that include the circle's center and extend to the circumference, along with all open regions outside the circle that extend to the circumference.

Any region of the plane (i.e., any set of points in space) for which if separated into two subsets it is possible to achieve that upon intersecting them (reuniting) the result is an empty set, but only at the cost of omitting some point from the original sets or returning to the original set, but only at the cost of not omitting any of the original points, so that their intersection would not be an empty set, is a connected set, a single piece, an indivisible totality.

2. Algorithm Implementation

2.1. Base Algorithm

ALGORITHM IN SPANISH

Input: A number of vertices of a graph G, Adjacency matrix of V(G).

Output: A topology of G (TG).

```
1. Insert the number of vertices of G.

2. for i \in n

Enter the name of the vertices of a graph G.

end

3. for i \in n

for j \in n

Enter the adjacency matrix of V(G).

end

end
```

```
4. for i \in n

for j \in n

Calculate the degree of V(G).

end

end

5. for i \in n

for j \in n

if (degree != 0)

class(i, j) = x(j).

R = (degree(x(i))x(t), degree(x(j))x(f)).

end

end

end

6. for i \in n

for j \in n

if (class(i, j) != 0)

subbase = class(j).

end

end

end

7. for v \in n

for vj \in n

base = subbase + intersection(v, vj).

end

end

8. for vi \in n

for vj \in n
```

```
if (union(v, vj) != (\emptyset))

vi, vj \in union.

union(vi, vj) \in union.

end

topology = base + union(vi, vj).

end

end
```

ALGORITHM IN ENGLISH

Input: A number of vertices of a graph G, Adjacency matrix of V(G).

Output: A topology of G (TG).

```
1. Insert the number of vertices of G.

2. for i \in n

Enter the name of the vertices of a graph G.

end

3.     for i \in n

for j \in n

Enter the adjacency matrix of V(G).

end

end

4.     for i \in n

for j \in n

Calculate the degree of V(G).

end

end

5.     for i \in n

for j \in n

if (degree != 0)

class(i, j) = x(j).
```

```
R = (degree(x(i))x(t), degree(x(j))x(f)).

end

end

end

6.     for i \in n

for j \in n

if (class(i, j) != 0)

subbase = class(j).

end

end

end

7.     for v \in n

for vj \in n

base = subbase + intersection(v, vj).

end

end

8.     for vi \in n

for vj \in n

if (union(v, vj) != (\emptyset))

vi, vj \in union.

union(vi, vj) \in union.

end

topology = base + union(vi, vj).

end

end
```

2.2. Implementation Considerations

Any code (in R or another language) constructed to operationalize the above algorithm in its simplest form must contain the following steps:

1. Generate the relevant numerical data with the given parameters.
2. Create a completely connected graph using the data indices as vertices.
3. Assign numerical values as attributes to the graph vertices.
4. Define the relationship R according to Definition 2.1 of Nada et al. (2018, p. 2), considering the numerical values.
5. Obtain the subbase by taking the post-classes (images of the relationship) of each vertex.
6. Generate the base by including the total space X, the empty set, the vertices, and their neighborhoods.
7. Construct the topology by taking unions of base elements, according to the laws of topological spaces.
8. Print the resulting R relationship, subbase, base, and topology.

NOTE: The R relationship is defined considering the numerical values associated with the vertices. The subbase, base, and topology are obtained by following the mathematical laws of topological spaces.

Obviously, the above code could be modified to consider neighborhoods of vertices and allow open sets containing multiple vertices, which will generate a richer topology, as constructed by Nada et al. (2018, p. 9).

However, despite the fact that this consideration of neighborhoods and multiple vertices is correct, the functions produce topologies that become exponentially larger as the data set increases slightly, which makes this consideration alone (without any optimization) unfeasible for sufficiently large sets. For example, for a set of 129 observations, its calculation becomes unfeasible, both computationally and in terms of the time required for the researcher to obtain valuable conclusions about the discovered topology[ˆ2].

The above problem occurs due to the way n-dimensional open balls are constructed[ˆ3] (i.e., the neighborhoods using the adjacency matrix), specifically when constructing the neighborhoods of each vertex with the adjacency matrix.

To optimize this process and make it viable for large sets, the following modifications are made:

1. Define a metric and a threshold for neighborhood creation. Thus, instead of connecting each vertex with all others, we can define a maximum distance to consider two points as neighbors.
2. Limit the size of the base. Specifically, a maximum number of elements in the base will be established, selecting the most representative ones.
3. Use a topology approximation. This will be achieved by using approximation methods that capture the essential structure of the topology without considering all possible unions (i.e., without generating all possible sets).

These criteria are mathematically valid and can be considered a robust form of approximating the original topology. However, it is important to note that this approximation may not capture all topological properties of the original space. Therefore, it might be recommended to use the syntax of the second code block (the immediately preceding block) for small sets (30 or fewer observations).

2.3. Optimizations and Heuristics

Since the definitions of *distance* and *threshold* are crucial in optimizing topology search, the methodological and theoretical-mathematical robustness of such optimization lies in the robustness of such definitions in the same senses.

Since we are working with a single real number variable, the distance to be defined is a one-dimensional distance.

Although defining the distance poses no complication, it is different when defining the radius of open balls (i.e., the threshold). For this, there are different possible heuristics, each with its advantages and disadvantages:

a. Average Distance between Adjacent Points

threshold <- mean(abs(diff(sort(data))))

This has the advantage of capturing the average scale of local variations in the data, but it can be sensitive to outliers and may not be suitable when the data does not have a location parameter that is the mean (as is the case with the Normal distribution).

b. Median of Adjacent Points Distances

threshold <- median(abs(diff(sort(data))))

The advantage of this estimation is that it is more robust to outliers than the average, but it has the disadvantage that it might not capture the structure well if there are many repeated values[^4].

c. Standard Deviation of the Data

threshold <- sd(data)

This heuristic would have the advantage of being able to capture the global dispersion of the data, however, it can be too large if there are extreme outliers and, consequently, become computationally and human time infeasible (which is precisely what we want to avoid).

d. Interquartile Range Divided by a Factor

threshold <- IQR(data) / factor

This heuristic has the advantage of being very robust to outliers, but could be too conservative if the data is very grouped[^5].

The division factor, for example, 4, is not a fixed rule. It is commonly used in statistics to identify outliers (1.5 * IQR is a common rule), but for our purpose of defining a threshold, we could adjust it. The general logic is that dividing by a larger number gives a smaller threshold, resulting in a finer topology.

In reality, the choice of factor depends on the level of detail desired in the topology.

A smaller factor (for example, dividing by 2 instead of 4) would result in a larger threshold, leading to a coarser topology with fewer sets. A larger factor (for example, dividing by 8) would result in a smaller threshold, leading to a finer topology with more sets.

The choice of factor should be based on experiments with the specific data and the level of detail needed for the analysis. There is no universal rule for this factor, as there is not for any heuristic. For this, the "analyze_topology_factors" function of this library should be used.

```
# IF LOCAL INSTALLATION IS DESIRED, THE FOLLOWING STEPS MUST BE FOLLOWED:
# Step 1:
setwd("/home/josemgomezj/Documentos/topologyR")
# Step 2:
library(devtools)
```

```
## Cargando paquete requerido: usethis
```

```
# Step 3:
library(roxygen2)
# Step 4:
document()
```

```
## i Updating topologyR documentation
```

```
## i Loading topologyR
```

```r
# Step 5:
install()
```

```
##
## -- R CMD build -------------------------------------------------------------
##      checking for file ‘/home/josemgomezj/Documentos/topologyR/DESCRIPTION’ ...  v  checking for file
##   -  preparing ‘topologyR’:
##    checking DESCRIPTION meta-information ...  v  checking DESCRIPTION meta-information
##   -  checking for LF line-endings in source and make files and shell scripts
##   -  checking for empty or unneeded directories
##   -  building ‘topologyR_0.1.0.tar.gz’
##
## Running /usr/lib64/R/bin/R CMD INSTALL /tmp/RtmpJaht6O/topologyR_0.1.0.tar.gz \
##   --install-tests
## * installing to library ‘/home/josemgomezj/R/x86_64-redhat-linux-gnu-library/4.4’
## * installing *source* package ‘topologyR’ ...
## ** using staged installation
## ** R
## ** byte-compile and prepare package for lazy loading
## ** help
## *** installing help indices
##   converting help for package ‘topologyR’
##     finding HTML links ... done
##     analyze_topology_factors                html
##     calculate_thresholds                    html
##     calculate_topology                      html
##     complete_topology                       html
##     is_topology_connected                   html
##     is_topology_connected2                  html
##     is_topology_connected_manual            html
##     simplest_topology                       html
##     visualize_topology_thresholds           html
## ** building package indices
## ** testing if installed package can be loaded from temporary location
## ** testing if installed package can be loaded from final location
## ** testing if installed package keeps a record of temporary installation path
## * DONE (topologyR)
```

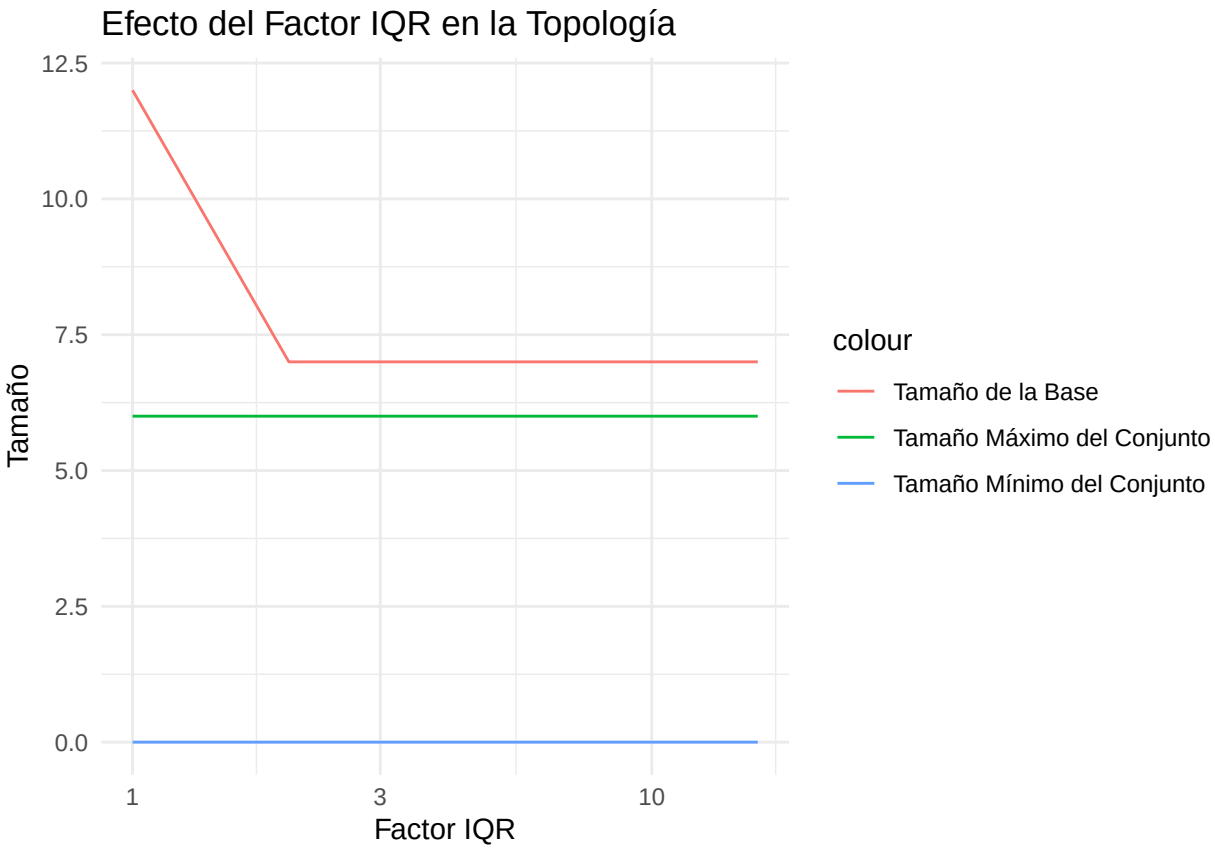
```r
# ONCE THE ABOVE STEPS ARE COMPLETED, FROM HERE ON YOU CAN PROCEED TO USE THE LIBRARY WITH LOCAL INSTAL
topology <- list(
  c(1, 2, 3),
  c(3, 4, 5)
)

# For the specific case of this function, check the structure of test_topology
str(topology)
```

```
## List of 2
##  $ : num [1:3] 1 2 3
##  $ : num [1:3] 3 4 5
```

```r
# If it's a list, try converting to a numeric vector
# For example:
test_topology_numeric <- unlist(topology)
```

```r
# Then try the function
analyze_topology_factors(test_topology_numeric)
```




```
##   factor threshold base_size max_set_size min_set_size
## 1      1   1.50000        12            6            0
## 2      2   0.75000         7            6            0
## 3      4   0.37500         7            6            0
## 4      8   0.18750         7            6            0
## 5     16   0.09375         7            6            0
```

The obtained graph will reveal what happens to the base size (number of sets in the topology) as the IQR factor increases. It is expected that the size will decrease, since a larger IQR factor implies a higher threshold for considering two elements as connected. Thus, with a higher threshold, fewer elements will be considered part of the same set, resulting in a simpler topology with fewer sets.

Regarding the maximum and minimum set sizes, it is natural that they do not change. The maximum size corresponds to the complete set of original elements, which will always be present in the topology, regardless of the IQR factor. The minimum size corresponds to the empty set, which is also part of the topology by definition. These extreme sets are not affected by the IQR factor.

Therefore, the expected effect of the IQR factor on the topology is that a larger IQR factor will generate a simpler topology, while the extreme sets (complete and empty) remain unchanged.

e. Density-Based Method (similar to that used in DBSCAN[^6])

k <- ceiling(log(length(data)))

sorted_distances <- sort(dist(matrix(data, ncol=1)))

threshold <- sorted_distances[k * length(data)]

This method has the advantage of adapting the threshold to the local density of the data, but its disadvantage is that it can be computationally more expensive for large data sets (which we are trying to avoid).

Without preliminary tests, among the recommended options, it would seem more appropriate to use either the median of distances between adjacent points or the density-based method. The reason is that these methods are robust to outliers and adapt well to the local structure of the data.

However, the most recommendable approach is to perform the pertinent tests to determine which of the heuristics in question is appropriate for each specific data set that the researcher has. Ultimately, the choice of heuristic will depend on the specific global topology properties to be studied. In this case, the desired property is connectivity.

3. Package Functions

3.1. is_topology_connected()

Purpose: Verifies if a topology is completely connected using an undirected graph approach. This function is particularly useful for analyzing the interconnection of data sets.

Functionality:

1. Converts the topology (list of sets) into an undirected adjacency matrix.
2. Performs a depth-first search (DFS) to explore connectivity between elements.
3. Determines if all topology elements can be reached from an initial point.

Usage Example:

```
topology <- list(c(1,2,3), c(3,4,5))
is_topology_connected(topology)  # Returns TRUE or FALSE
```

```
## [1] TRUE
```

Special Cases:

1. Returns FALSE if the topology is empty.
2. Handles topologies with dispersed or non-consecutive elements.

3.2. is_topology_connected2()

Purpose: Similar to is_topology_connected, but uses a directed graph approach, allowing for more specific analyses of directional relationships.

Functionality:

1. Creates a directed adjacency matrix based on the order of elements in each set.
2. Performs a depth-first search (DFS) to verify connectivity.
3. Focuses on sequential connections between elements.

Usage Example:

```
topology <- list(c(1,2,3), c(3,4,5))
is_topology_connected2(topology)  # Returns TRUE or FALSE
```

```
## [1] TRUE
```

Key Difference from is_topology_connected:

1. Considers the directions of connections between elements.
2. More restrictive in terms of connectivity.

3.3. is_topology_connected_manual()

Purpose: Provides a manual and direct method for verifying topology connectivity.

Functionality:

1. Checks if all original elements (from 1 to the maximum value) are present in at least one topology set.
2. Useful for topologies with specific structure or particular constraints.

Usage Example:

```
topology <- list(c(1,2,3), c(3,4,5))
is_topology_connected_manual(topology)  # Returns TRUE or FALSE
```

```
## [1] TRUE
```

Special Use Case: Ideal for topologies with specific completeness requirements.

3.4. analyze_topology_factors()

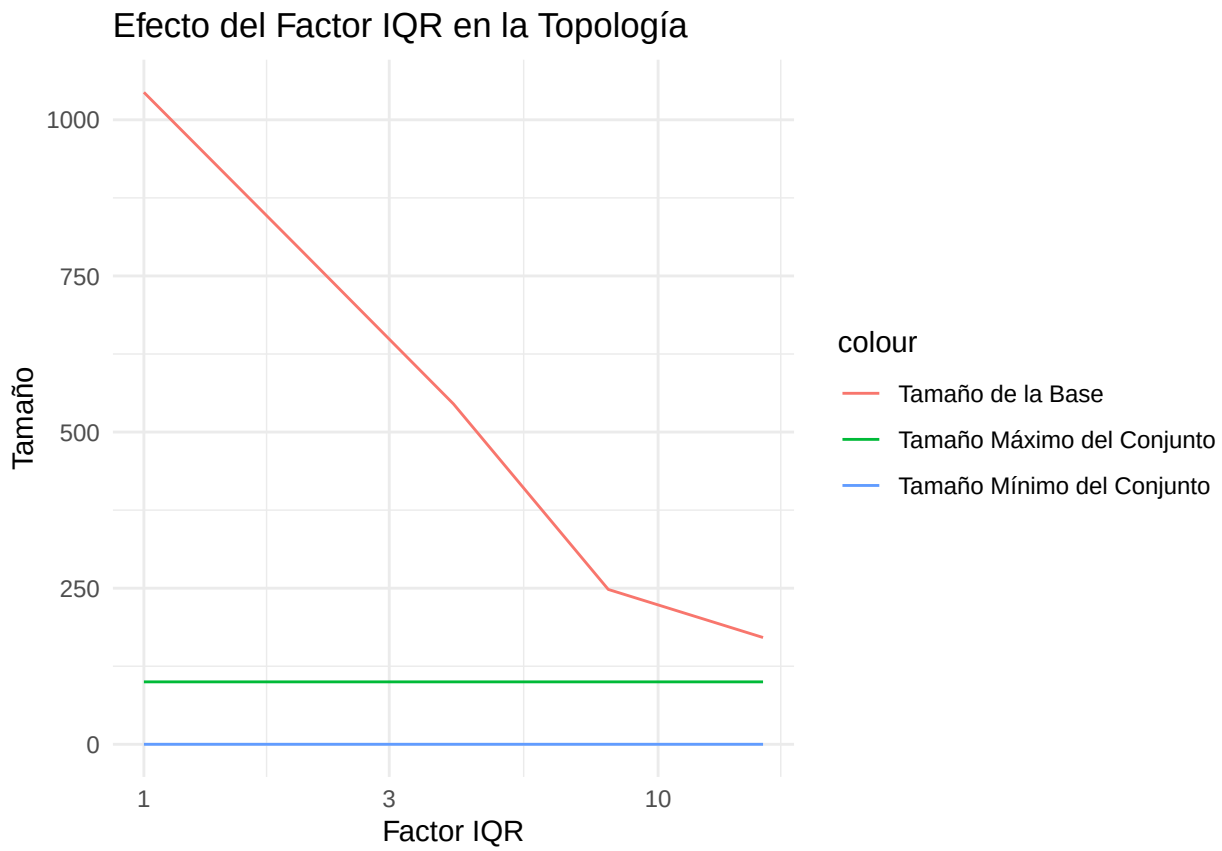
Purpose: Analyzes how different Interquartile Range (IQR) factors affect topology characteristics.

Functionality:

1. Tests multiple IQR factors (default: 1, 2, 4, 8, 16).
2. Calculates thresholds, generates subbases and topological base for each factor.
3. Produces a summary of topology characteristics.
4. Optionally generates a graphical visualization.

Usage Example:

```
data <- rnorm(100)
topologyR:::analyze_topology_factors(data)
```




```
##   factor threshold base_size max_set_size min_set_size
## 1      1 1.2951536      1044          100            0
## 2      2 0.6475768       795          100            0
## 3      4 0.3237884       545          100            0
## 4      8 0.1618942       248          100            0
## 5     16 0.0809471       171          100            0
```

```
# IQR factor analysis with included graph
```

Key Features:

1. Helps choose the optimal IQR factor for a data set.
2. Provides information about the topological base size.

3.5. calculate_thresholds()

Purpose: Calculates multiple thresholds for topological analysis using different statistical methods.

Calculation Methods:

1. Mean difference
2. Median difference
3. Standard deviation
4. Adjusted IQR
5. DBSCAN method

Usage Example:

```r
data <- c(1, 2, 3, 4, 5)
topologyR:::calculate_thresholds(data)
```

```
## $mean_diff
## [1] 1
##
## $median_diff
## [1] 1
##
## $sd
## [1] 1.581139
##
## $iqr
## [1] 0.5
##
## $dbscan
## [1] 4
```

```r
# Returns a list with different thresholds
```

Utility: Provides multiple perspectives for defining thresholds in topological analysis.

3.6. visualize_topology_thresholds()

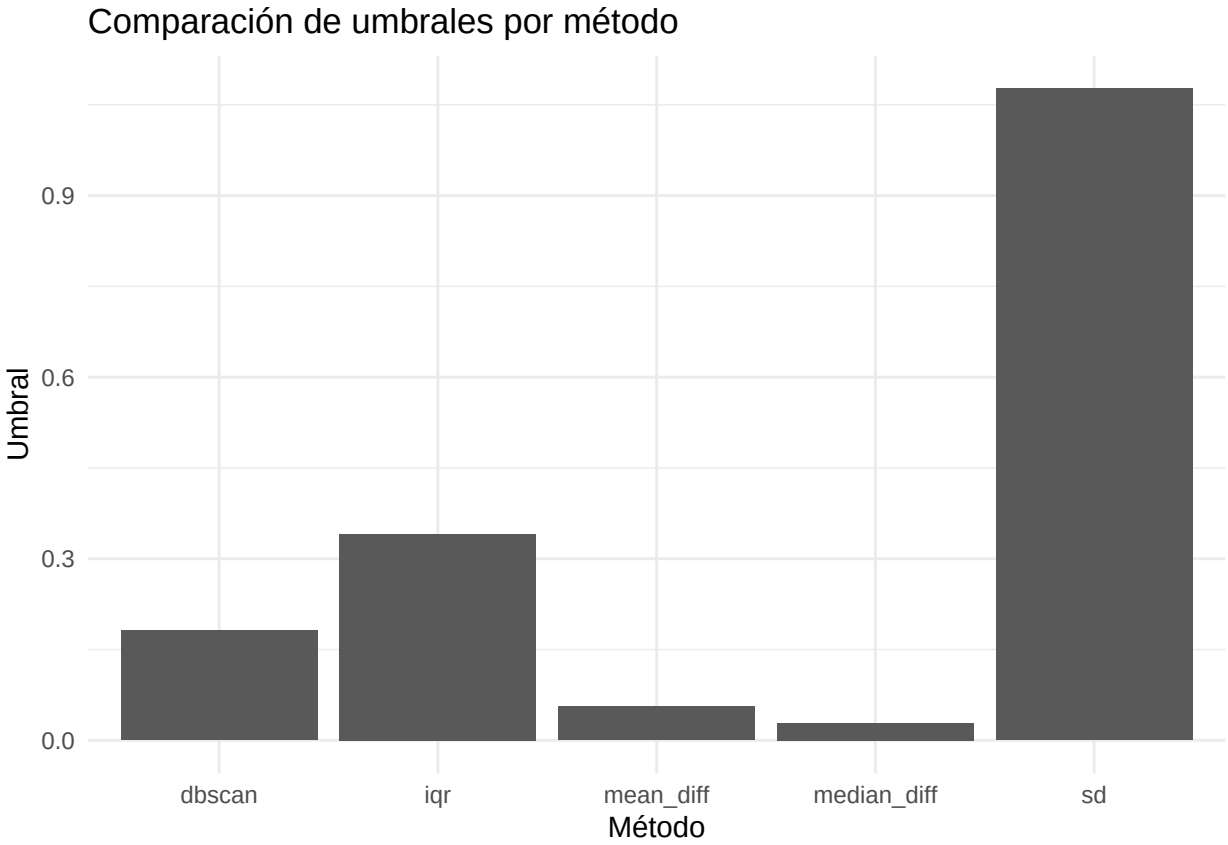
Purpose: Visualizes and compares calculated thresholds and their effects on topological structure.

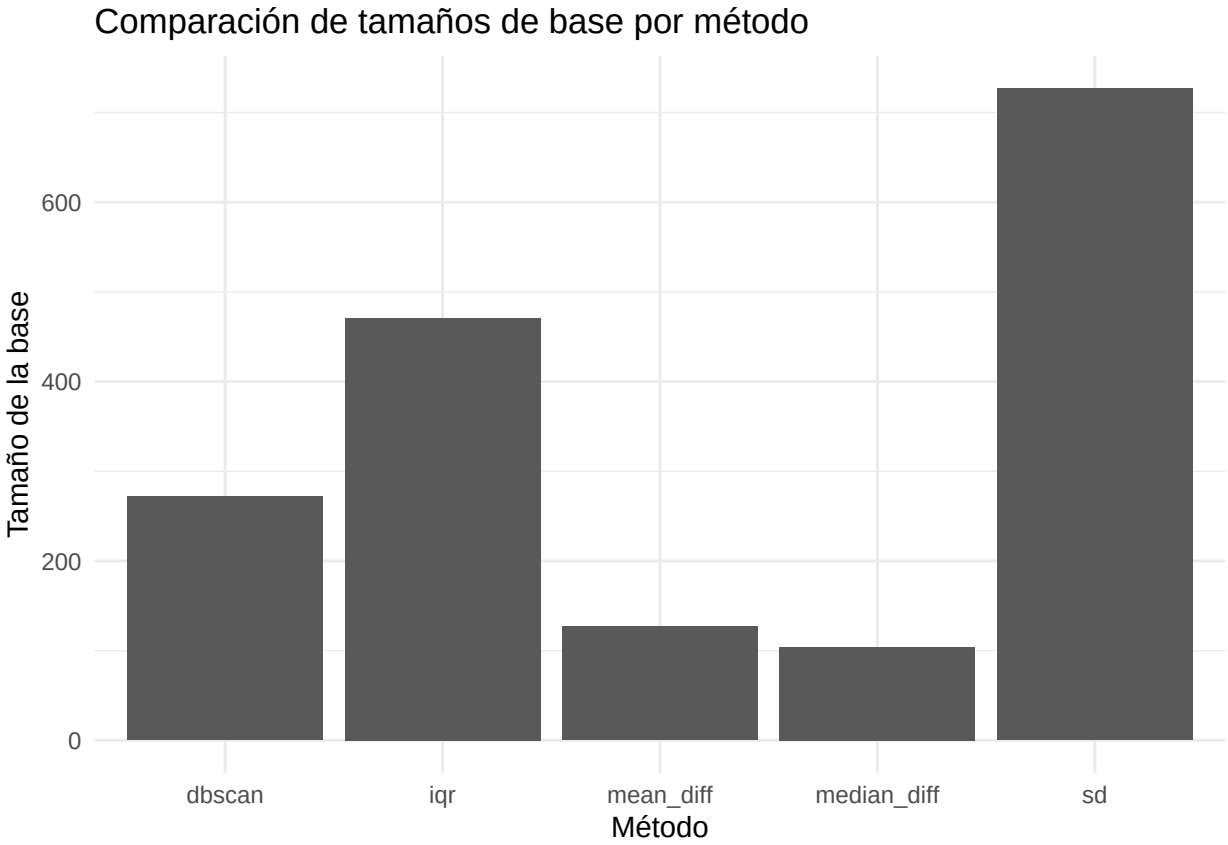
Functionality:

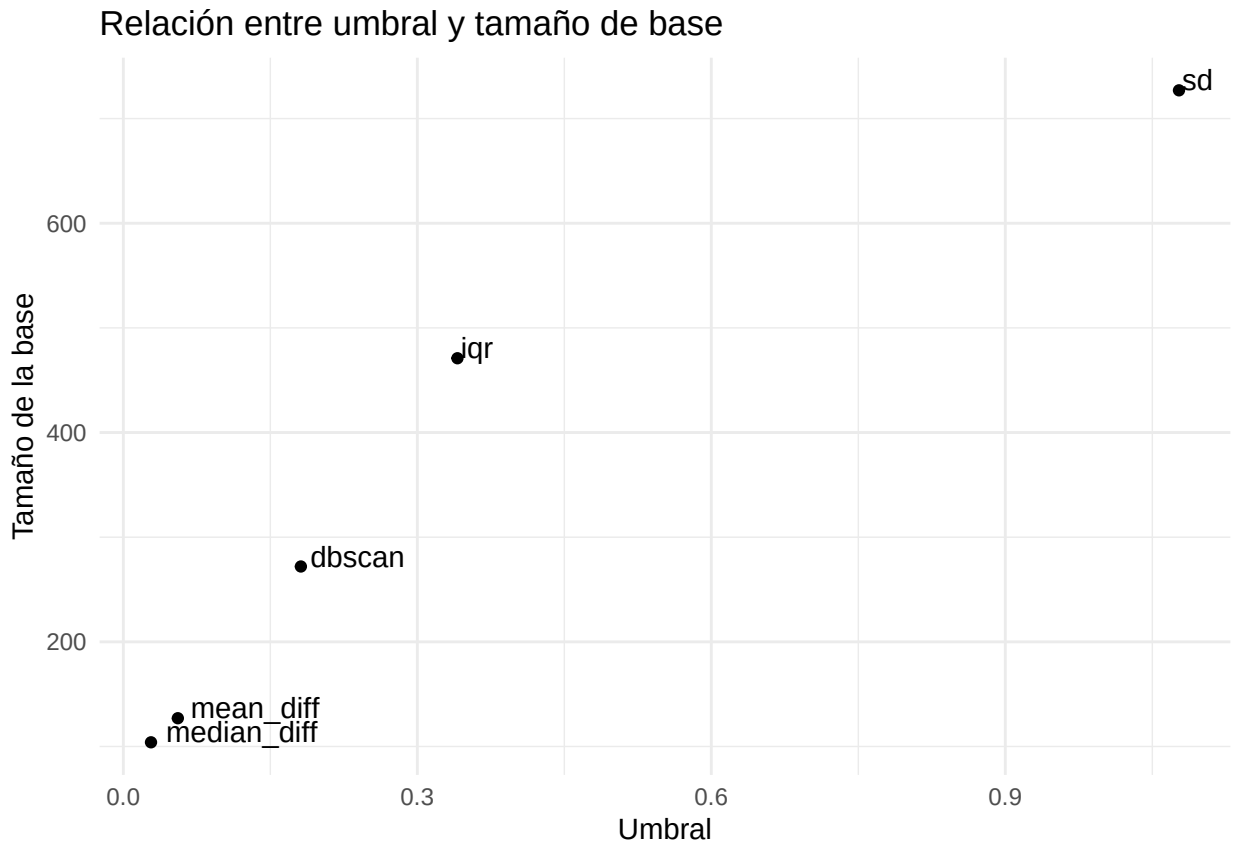
1. Calculates thresholds using calculate_thresholds
2. Generates comparative graphs: 2.1. Threshold comparison by method 2.2. Base sizes by method 2.3. Relationship between threshold and base size

Usage Example:

```r
data <- rnorm(100)
topologyR:::visualize_topology_thresholds(data, plot = TRUE)
```


Comparación de umbrales por método
Umbral
0.0
0.3
0.6
0.9
dbscan
iqr
mean_diff
median_diff
sd
Método


Comparación de tamaños de base por método
Tamaño de la base
600
400
200
0
dbscan
iqr
mean_diff
median_diff
sd
Método




```
##                  method  threshold base_size
## mean_diff     mean_diff 0.05562107       127
## median_diff median_diff 0.02824408       104
## sd                   sd 1.07720228       727
## iqr                 iqr 0.34093081       471
## dbscan           dbscan 0.18117414       272
```

```
# Generates three graphs and returns a data frame
```

Key Features:

1. Interactive visualization of topological characteristics
2. Helps understand the impact of different thresholding methods

4. Applications

4.1. Economic Time Series Analysis

The topologyR package provides specialized tools for analyzing economic time series data through topological methods. This section demonstrates practical applications using real GDP data.

4.1.1. Data Preparation and Distribution Analysis

When analyzing economic time series data, particularly GDP changes, the distribution choice is crucial for topology construction for large datasets when we are facing restrictions in computational resources.

However, before determining the optimal distribution of our data we must determine the optimal method to empirically fit our data to theoretical distributions or, to be more precise, to empirical distributions that

perfectly follow a theoretical distribution.

MLE (Maximum Likelihood Estimation)

When to Use:

- Large samples with correct parametric model specification.
- Prioritize statistical efficiency (minimum asymptotic variance).
- Data is well-behaved (no heavy tails/outliers; e.g., normal distribution).
- Known distributional form (parametric assumptions hold).

Avoid: Small samples, misspecified models, or heavy-tailed distributions.

Example: Fitting a normal distribution to clean, symmetric data.

MME (Moment Matching Estimation)

When to Use:

- Small samples or computational simplicity required.
- Distributional form uncertain, but moments are calculable/stable.
- Real-time applications prioritizing speed over precision.
- Avoid for distributions with undefined moments (e.g., Cauchy).

Example: Quick parameter estimation for gamma/beta distributions.

QME (Quantile Matching Estimation)

When to Use:

- Focus on tail behavior/quantiles (e.g., VaR in finance).
- Robustness to outliers/censored data (quantiles are less sensitive).
- Asymmetric/heavy-tailed distributions (e.g., Pareto for extreme losses).

Example: Modeling insurance claim extremes using the 95th percentile.

MGE (Maximum Goodness-of-Fit Estimation)

When to Use:

- Validating distributional hypotheses (e.g., KS/AD tests).
- Comparing multiple distributions globally.
- Non-standard distributions where MLE/MME fail.

Example: Testing if data follows logistic vs. Gumbel distributions.

MSE (Maximum Spacing Estimation)

When to Use:

Data with gaps/rounding (e.g., irregular measurements).

Heavy-tailed distributions or small continuous samples.

Avoids overreliance on extreme values (spacing-based robustness).

Example: Fitting a Weibull distribution to irregular failure-time data.

```r
library(knitr)
estimation_methods <- data.frame(
  Method = c("MLE", "MME", "QME", "MGE", "MSE"),
  When_To_Use = c(
    "Large samples, correct model, efficiency priority",
    "Small samples, speed priority, moments exist",
    "Tail/quantile focus, robustness to outliers",
    "Distribution validation/comparison, non-standard cases",
    "Gapped data, heavy tails, small samples"
  ),
  Key_Advantages = c(
    "Asymptotic efficiency, parametric precision",
    "Computational simplicity, no likelihood needed",
    "Robust quantile alignment, tail accuracy",
    "Global fit assessment, hypothesis testing",
    "Robust to gaps/outliers, spacing consistency"
  ),
  R_Functions = c(
    "`fitdistr` (MASS), `mle` (stats4)",
    "`optim` + manual moment equations",
    "`qme` (custom), `quantreg::rq`",
    "`goftest`, `ks.test`, `ad.test`",
    "`mps` (POT), `MPS.est` packages"
  )
)

colnames(estimation_methods) <- gsub("_", " ", colnames(estimation_methods))

kable(
  estimation_methods,
  format = "pipe",
  caption = "**ESTIMATION METHOD BENCHMARK**",
  align = c("c", "c", "c", "c")  # Centrar todas las columnas
)
```

Table 1: **ESTIMATION METHOD BENCHMARK**

Method	When To Use	Key Advantages	R Functions
MLE	Large samples, correct model, efficiency priority	Asymptotic efficiency, parametric precision	`fitdistr` (MASS), `mle` (stats4)
MME	Small samples, speed priority, moments exist	Computational simplicity, no likelihood needed	`optim` + manual moment equations
QME	Tail/quantile focus, robustness to outliers	Robust quantile alignment, tail accuracy	`qme` (custom), `quantreg::rq`

Method	When To Use	Key Advantages	R Functions
MGE	Distribution validation/comparison, non-standard cases	Global fit assessment, hypothesis testing	`goftest`, `ks.test`, `ad.test`
MSE	Gapped data, heavy tails, small samples	Robust to gaps/outliers, spacing consistency	`mps` (POT), `MPS.est` packages

We are going to provide users of this library with a custom function to fit multiple types of distributions simultaneously. To do this, we will provide the user with data on US quarterly GDP growth from the first quarter of 1992 to the first quarter of 2024 after applying a Yeo-Johnson transformation to the data (with a value for the hyperparameter λ equal to 0.5).

```
X <- c(
  0.0119368777374702, 0.0108137656182881, 0.00985681082011425, 0.0104015519293652,
  0.0016687038568594 7, 0.00581325152667178, 0.00476582173579576, 0.0135555616868386,
  0.00968017355996231, 0.0135061956696334, 0.005837281536067, 0.0114230783204214,
  0.00354465884841204, 0.00298097844188216, 0.00848928301845797, 0.00677931023817369,
  0.00747652539201571, 0.0166104234581352, 0.00895027315262364, 0.0103568837398003,
  0.00644401865589073, 0.0165834473187565, 0.0124469682453747, 0.00852045048090089,
  0.0100124377724629, 0.00923557603383074, 0.0125500242230006, 0.0160277775864102,
  0.00937233981161389, 0.00833005255610297, 0.0132151400185725, 0.016339951496275,
  0.00362461554054549, 0.0181307192548257, 0.00101904038917144, 0.00596151508447429,
  -0.00328068485517437, 0.00623528031983334, -0.00401041013477036, 0.00274591498772025,
  0.00834618529774378, 0.00611774330421611, 0.00406097711621545, 0.00123571825010149,
  0.0052594844558147, 0.0088366782792475, 0.0165634133346764, 0.011573712295923,
  0.00565849535757224, 0.00773454420647957, 0.00946351049229044, 0.0101762111814976,
  0.0110630024939544, 0.00492014803582652, 0.00782280094633858, 0.00554750629348089,
  0.0134088506808547, 0.00258582837290655, 0.00149903822110309, 0.00857611257328283,
  0.00300614077940331, 0.00610827225252253, 0.00575202854191303, 0.00627256373604412,
  -0.00427214986914152, 0.00594546286782194, -0.00525949142529332, -0.0220096627946815,
  -0.0113808376042674, -0.00178729765814796, 0.00350882204196923, 0.0107800476432023,
  0.0048391456673027, 0.00965260679551294, 0.0076959929232312, 0.00524422452727702,
  -0.00237340604041976, 0.00675439453860394, -0.000223112442939784, 0.0111991447889985,
  0.00836839250173416, 0.00445833082157154, 0.001439781757123, 0.00115606587792128,
  0.00984138677657853, 0.0026746116131795, 0.0084958551114811, 0.0086984840936184,
  -0.00345397563998976, 0.0128769460650098, 0.0121171934059903, 0.00505092204661972,
  0.00898461915466164, 0.00618453787282514, 0.00399850299345283, 0.00184394996213433,
  0.00578682815497622, 0.0032086261795099, 0.00708136357248845, 0.00553534000276334,
  0.00486298783732364, 0.00559088549983189, 0.00786971688902183, 0.01123892165998,
  0.00811762603688138, 0.00530127412316794, 0.00622730516758763, 0.00141589880764181,
  0.00542364601597312, 0.00827956221239301, 0.0112899343456179, 0.00640295055604412,
  -0.0136749297466007, -0.0804470145766672, 0.0761427696572317, 0.0103255457761064,
  0.0128141493938281, 0.01514019363418, 0.00812718720702543, 0.0168963285206307,
  -0.0049829870079526, -0.0014131988129115, 0.00657539105810834, 0.00634413797832822,
  0.00555608248684969, 0.00510468554636834, 0.011903277993254, 0.00836650041769005,
  0.00394730469640825
) # Real GDP Growth

library(fitdistrplus)
```

```
## Cargando paquete requerido: MASS
```

```
## Cargando paquete requerido: survival
```

```r
library(dplyr)
```

```
##
## Adjuntando el paquete: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Example of distribution fitting
x <- X  # Vector of GDP changes
distributions <- c("norm", "cauchy", "logis", "unif", "t")
fits <- list()

set.seed(100000)
for(dist in distributions) {
  if(dist == "t") {
    fits[[dist]] <- fitdist(x, dist, start=list(df=length(x)-1), method="mge")
  } else {
    fits[[dist]] <- fitdist(x, dist, method="mge")
  }
}
```

```
## Warning in fitdist(x, dist, method = "mge"): maximum GOF estimation has a
## default 'gof' argument set to 'CvM'
## Warning in fitdist(x, dist, method = "mge"): maximum GOF estimation has a
## default 'gof' argument set to 'CvM'
## Warning in fitdist(x, dist, method = "mge"): maximum GOF estimation has a
## default 'gof' argument set to 'CvM'
## Warning in fitdist(x, dist, method = "mge"): maximum GOF estimation has a
## default 'gof' argument set to 'CvM'
```

```
## Warning in fitdist(x, dist, start = list(df = length(x) - 1), method = "mge"):
## maximum GOF estimation has a default 'gof' argument set to 'CvM'
```

```r
fits$norm$estimate
```

```
##        mean          sd
## 0.006940808 0.004746252
```

```r
fits$norm$bic
```

```
## [1] -381.9854
```

```r
fits$cauchy$estimate
```

```
##    location       scale
## 0.006903160 0.002882685
```

```r
fits$cauchy$bic
```

```
## [1] -922.4761
```

```
fits$logis$estimate
```

```
##    location       scale
## 0.006937262 0.002859244
```

```
fits$logis$bic
```

```
## [1] -861.6542
```

```
fits$unif$estimate
```

```
##          min          max
## 0.0000480909 0.0138884505
```

```
fits$unif$bic
```

```
## [1] Inf
```

```
fits$t$estimate
```

```
##  df
## 128
```

```
fits$t$bic
```

```
## [1] 242.4721
```

```
library(topologyR)

X2 = rcauchy(length(X), 0.006903160, 0.002882685)

X3 = rcauchy(10, 0.006903160, 0.002882685)
```

4.1.2. Basic Topology Analysis

For datasets with size equal or less to 30 observations:

```
# Create initial topology
topology <- complete_topology(X3)
length(topology$R)
```

```
## [1] 100
```

```
length(topology$subbase)
```

```
## [1] 10
```

```
length(topology$base)
```

```
## [1] 57
```

```
length(topology$topology)
```

```
## [1] 190
```

```
# Check connectivity using different methods
is_connected_undirected <- is_topology_connected(topology$topology)
is_connected_directed <- is_topology_connected2(topology$topology)
is_connected_manual <- is_topology_connected_manual(topology$topology)

is_connected_undirected
```

```
## [1] TRUE
```

```
is_connected_directed
```

```
## [1] TRUE
```

```
is_connected_manual
```

```
## [1] TRUE
```

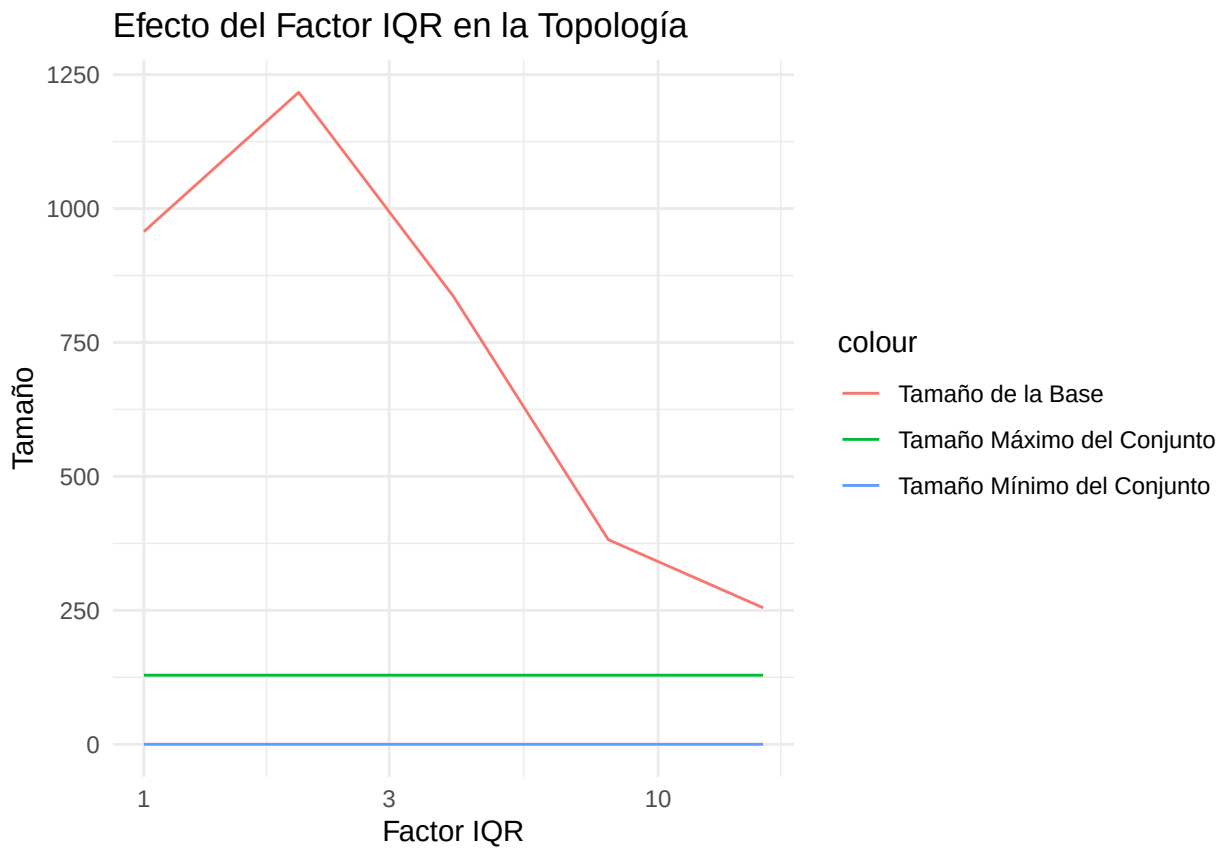
4.1.3. Advanced Analysis with Thresholds

For larger datasets ($n > 30$):

```
# Calculate optimal thresholds
thresholds <- calculate_thresholds(X)
thresholds
```

```
## $mean_diff
## [1] 0.001223358
##
## $median_diff
## [1] 0.0001172427
##
## $sd
## [1] 0.01148499
##
## $iqr
## [1] 0.001503484
##
## $dbscan
## [1] 0.0006469434
```

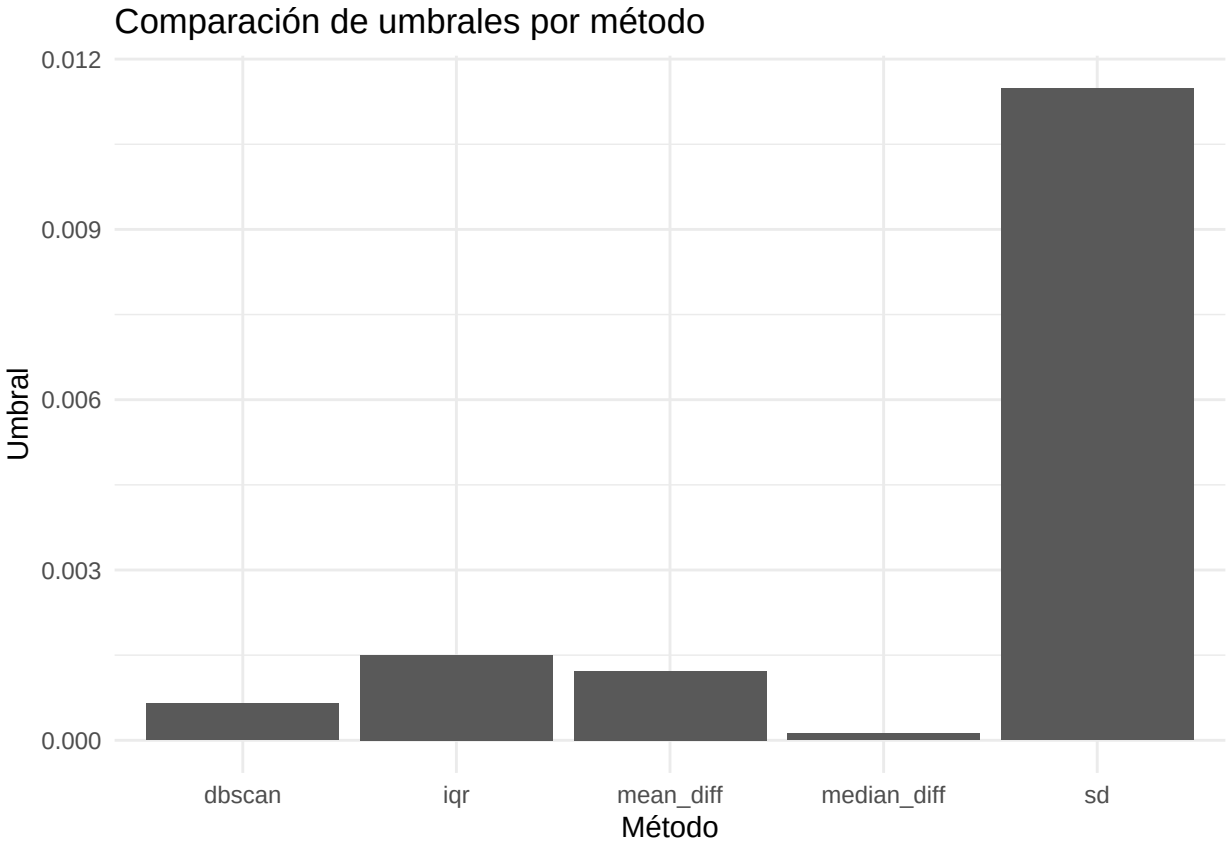
```
# Analyze topology characteristics with different IQR factors
results <- analyze_topology_factors(X, factors = c(1, 2, 4, 8, 16))
```

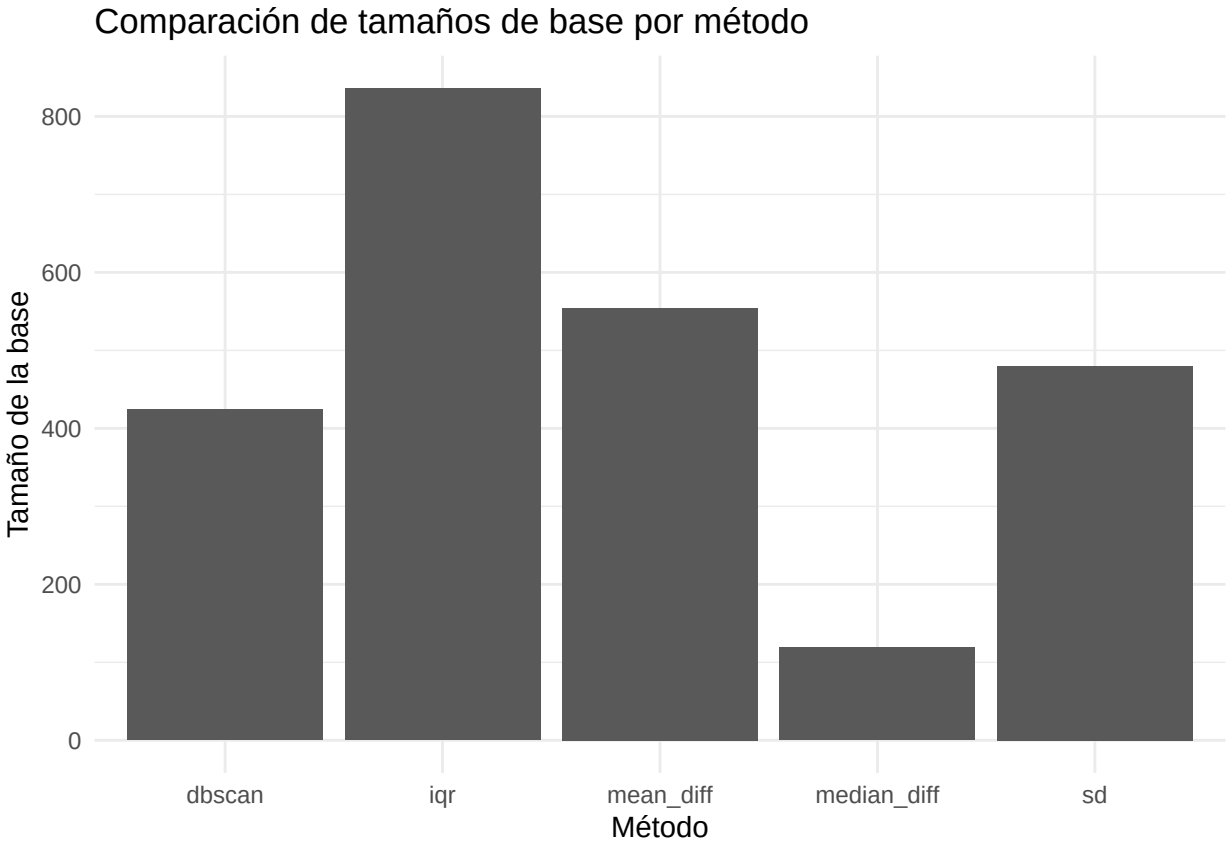


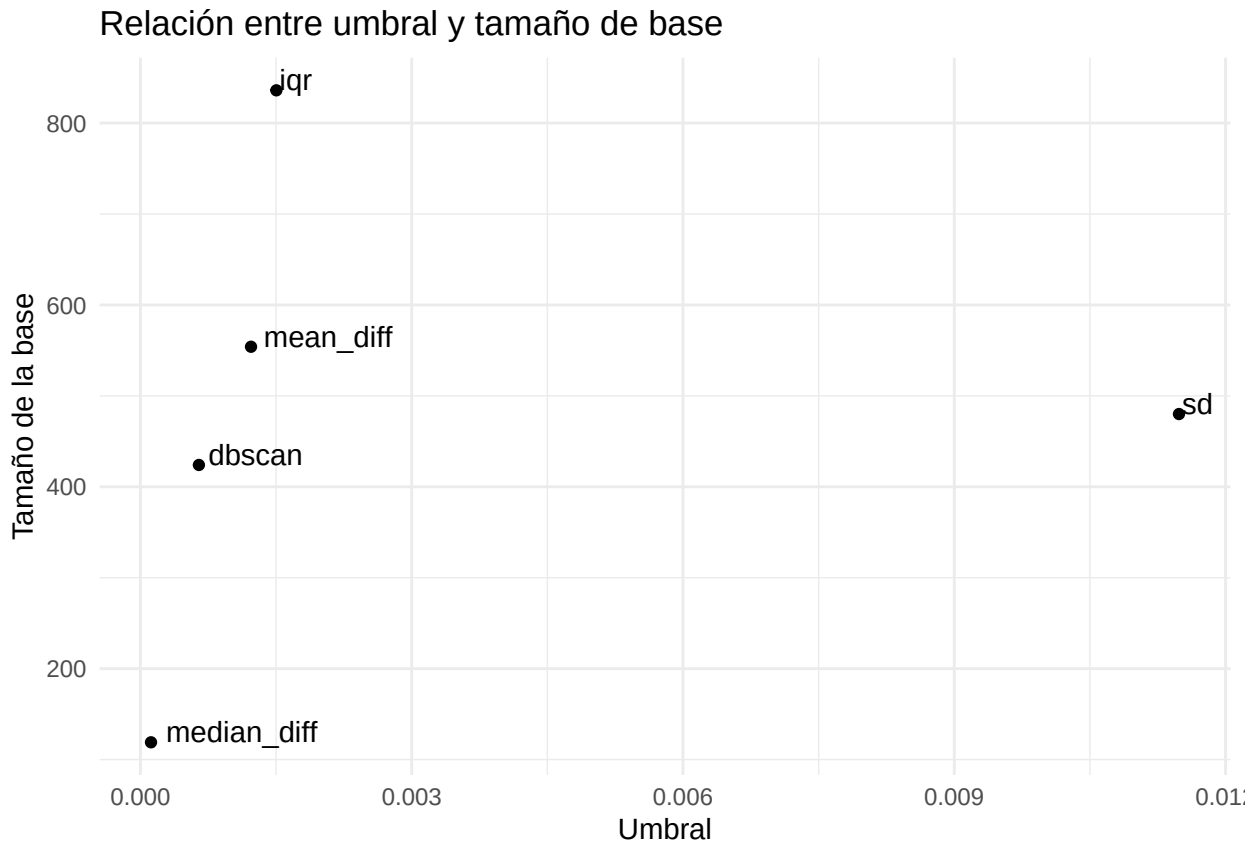

```
results
```

```
##   factor    threshold base_size max_set_size min_set_size
## 1      1 0.0060139348       957          129            0
## 2      2 0.0030069674      1217          129            0
## 3      4 0.0015034837       836          129            0
## 4      8 0.0007517418       382          129            0
## 5     16 0.0003758709       255          129            0
```

```
# Visualize threshold comparisons
viz_results <- visualize_topology_thresholds(X)
```


Comparación de umbrales por método
0.012
0.009
0.006
0.003
0.000
Umbral
dbscan
iqr
mean_diff
median_diff
sd
Método


Comparación de tamaños de base por método
Tamaño de la base
800
600
400
200
0
dbscan
iqr
mean_diff
median_diff
sd
Método




```
viz_results
```

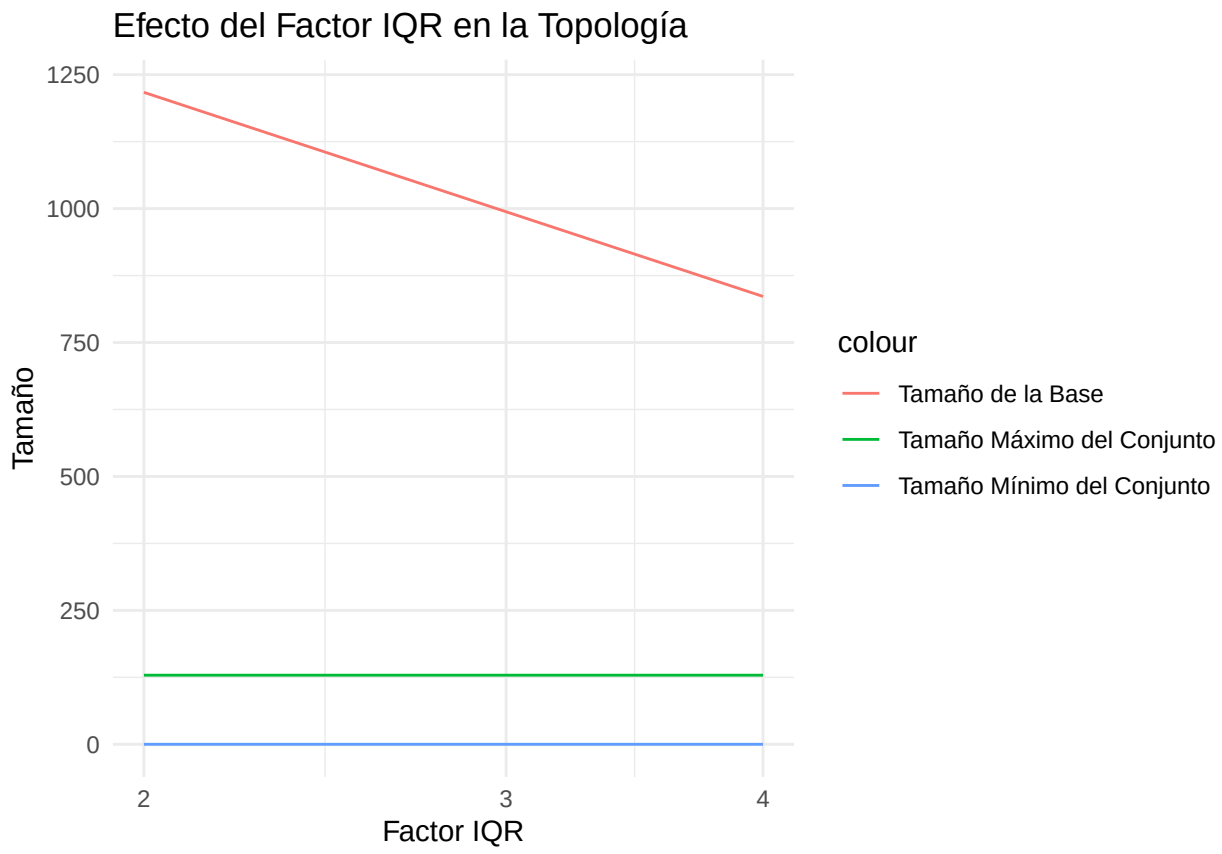
```
##                  method    threshold base_size
## mean_diff     mean_diff 0.0012233577       554
## median_diff median_diff 0.0001172427       119
## sd                   sd 0.0114849870       480
## iqr                 iqr 0.0015034837       836
## dbscan           dbscan 0.0006469434       424
```

4.1.4. Performance Optimization Guidelines

Dataset Size Recommendations

- $n \leq 30$: Use `complete_topology()`.
- $30 < n \leq 100$: Use threshold-based methods with default parameters.
- $n > 100$: Use optimized thresholds:

```
threshold <- IQR(x)/4  # Default conservative threshold
results <- analyze_topology_factors(x, factors = c(2, 4))
```



4.1.5. Interpretation Framework

1. Connectivity Analysis

- Connected topology: Stable economic period.
- Disconnected topology: Potential structural breaks.

2. Base Size Interpretation

```
base_size <- length(topology$base)
relative_size <- base_size/length(x)
```

- relative_size < 0.3: Simple structure.Based on Kolmogorov complexity theory, a topology with fewer sets than 30% of possible combinations indicates high data compressibility. This corresponds to strong regularity in the underlying economic dynamics, which aligns with minimum description length principle, suggesting a parsimonious representation.
- $0.3 \leq$ relative_size ≤ 0.7: Moderate complexity. Derived from entropy considerations in dynamical systems, reflects balance between order and disorder typical of stable economic systems. This range corresponds to phase transition region in complex networks and is supported by empirical studies showing most stable economic periods exhibit topological bases in this range.
- relative_size > 0.7: Complex structure. Draws from random graph theory, in which the high relative size indicates near-random structure. This suggests potential market inefficiencies or structural instabilities and corresponds to high algorithmic complexity in the sense of Chaitin. Is typically observed during periods of economic turbulence or systemic change.

The above considerations must not be confused with complexity in the sense of complex systems/chaos theory.

Topological complexity measures the richness of the neighborhood structure and connectedness patterns. A high relative_size indicates many distinct open sets and intricate local relationships. Chaos-theory complexity measures the system's sensitivity to initial conditions and long-term predictability. A chaotic system can have simple topology (few open sets) but complex dynamics.

For example, the logistic map can exhibit chaos with a relatively simple topological structure (relative_size < 0.3), while a stable periodic system might require complex topology (relative_size > 0.7) to capture its neighborhood relationships.

4.2. Comparative Analysis Examples

```
#Real GDP Growth quarterly data from 1960-04-01 to 2024-01-01 (FRED Data)
x <- c(
  2.2, -0.5, 0.5, -1.3, 0.7, 1.7, 1.9, 2.0, 1.8, 0.9, 1.2, 0.3, 1.1, 1.1, 2.2,
  0.7, 2.1, 1.1, 1.6, 0.3, 2.4, 1.3, 2.2, 2.3, 2.4, 0.3, 0.8, 0.8, 0.9, 0.1, 0.9,
  0.8, 2.0, 1.7, 0.8, 0.4, 1.6, 0.3, 0.7, -0.5, -0.1, 0.1, 0.9, -1.1, 2.7, 0.5, 0.8,
  0.2, 1.8, 2.3, 0.9, 1.7, 2.5, 1.1, -0.5, 0.9, -0.9, 0.2, -0.9, -0.4, -1.2, 0.7,
  1.7, 1.3, 2.2, 0.7, 0.5, 0.7, 1.2, 1.9, 1.8, 0.0, 0.3, 3.9, 1.0, 1.3, 0.2, 0.1,
  0.7, 0.3, 0.3, -2.1, -0.1, 1.9, 2.0, -0.7, 1.2, -1.1, -1.6, 0.5, -0.4, 0.0, 1.3,
  2.3, 2.0, 2.1, 2.0, 1.7, 1.0, 0.8, 1.0, 0.9, 1.5, 0.7, 0.9, 0.5, 1.0, 0.5, 0.7,
  1.1, 0.9, 1.7, 0.5, 1.3, 0.6, 1.3, 1.0, 0.8, 0.7, 0.2, 1.1, 0.4, 0.1, -0.9, -0.5,
  0.8, 0.5, 0.3, 1.2, 1.1, 1.0, 1.0, 0.2, 0.6, 0.5, 1.4, 1.0, 1.4, 0.6, 1.1, 0.4,
  0.3, 0.9, 0.7, 0.7, 1.7, 0.9, 1.0, 0.6, 1.7, 1.2, 0.9, 1.0, 0.9, 1.3, 1.6, 0.9,
  0.8, 1.3, 1.6, 0.4, 1.8, 0.1, 0.6, -0.3, 0.6, -0.4, 0.3, 0.8, 0.6, 0.4, 0.1, 0.5,
  0.9, 1.7, 1.2, 0.6, 0.8, 0.9, 1.0, 1.1, 0.5, 0.8, 0.6, 1.3, 0.3, 0.1, 0.9, 0.3,
  0.6, 0.6, 0.6, -0.4, 0.6, -0.5, -2.2, -1.1, -0.2, 0.4, 1.1, 0.5, 1.0, 0.8, 0.5,
  -0.2, 0.7, 0.0, 1.1, 0.8, 0.4, 0.1, 0.1, 1.0, 0.3, 0.9, 0.9, -0.3, 1.3, 1.2, 0.5,
  0.9, 0.6, 0.4, 0.2, 0.6, 0.3, 0.7, 0.6, 0.5, 0.6, 0.8, 1.1, 0.8, 0.5, 0.6, 0.1,
  0.5, 0.8, 1.1, 0.6, -1.4, -7.9, 7.8, 1.0, 1.3, 1.5, 0.8, 1.7, -0.5, -0.1, 0.7,
  0.6, 0.6, 0.5, 1.2, 0.8, 0.4
)


# Compare multiple time periods
periods <- list(
  pre_crisis = x[1:39],
  various_crisis = x[40:125],
  post_crisis = x[126:192],
  all_period = x
)

# Data quality check
for(name in names(periods)) {
  missing <- sum(is.na(periods[[name]]))
  if(missing > 0) {
    warning(sprintf("Period %s has %d missing values", name, missing))
  }
}

results <- lapply(periods, function(period_data) {
  if(length(period_data) <= 30) {
    topology <- complete_topology(period_data)
  } else {
    # Use threshold-based approach
```

```
    threshold <- IQR(period_data)/4
    topology <- calculate_topology(period_data, threshold)
  }
  list(
    connectivity = is_topology_connected(topology),
    complexity = length(topology)/length(period_data)
  )
})

results
```

```
## $pre_crisis
## $pre_crisis$connectivity
## [1] TRUE
##
## $pre_crisis$complexity
## [1] 0.02564103
##
##
## $various_crisis
## $various_crisis$connectivity
## [1] TRUE
##
## $various_crisis$complexity
## [1] 0.01162791
##
##
## $post_crisis
## $post_crisis$connectivity
## [1] TRUE
##
## $post_crisis$complexity
## [1] 0.01492537
##
##
## $all_period
## $all_period$connectivity
## [1] TRUE
##
## $all_period$complexity
## [1] 0.003891051
```

4.3. Some Recommendations for Your Dataset

4.3.1. Data Preparation

```
# Crear un vector numérico de juguete con algunos NA's
dummy_var <- c(1.5, 2.3, NA, 4.7, 5.1, NA, 3.2, 6.4, NA, 7.8)

prepare_data <- function(x) {
  # Remove missing values
  x <- na.omit(x)
  # Scale if necessary
  if(sd(x) > 1) x <- scale(x)
```

```
  # Check minimum sample size
  if(length(x) < 10) warning("Sample size may be too small")
  x
}

prepared_data <- prepare_data(dummy_var)
```

```
## Warning in prepare_data(dummy_var): Sample size may be too small
```

```
print(prepared_data)  # Shows scaled values without NAs
```

```
##            [,1]
## [1,] -1.3011743
## [2,] -0.9457316
## [3,]  0.1205966
## [4,]  0.2983180
## [5,] -0.5458585
## [6,]  0.8759125
## [7,]  1.4979372
## attr(,"scaled:center")
## [1] 4.428571
## attr(,"scaled:scale")
## [1] 2.250714
```

4.3.2. Error Handling:

```
safe_topology_analysis <- function(x) {
  tryCatch({
    threshold <- IQR(x)/4
    topology <- calculate_topology(x, threshold)
    list(
      topology = topology,
      connectivity = is_topology_connected(topology),
      base_size = length(topology)
    )
  }, error = function(e) {
    message("Error in topology analysis: ", e$message)
    NULL
  })
}

result <- safe_topology_analysis(X3)
print(str(result))
```

```
## List of 3
##  $ topology    : int 8
##  $ connectivity: logi TRUE
##  $ base_size   : int 1
## NULL
```

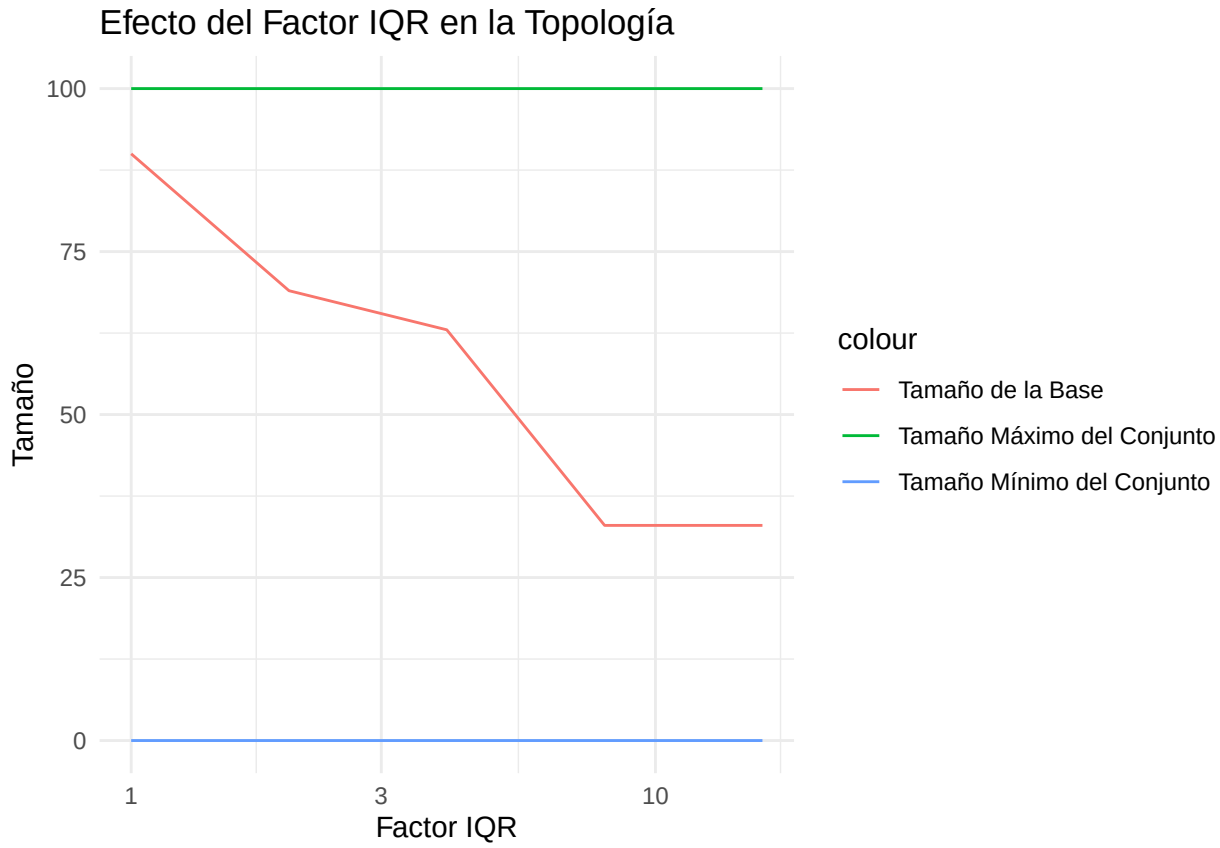
4.3.3. Optimization for Large Datasets

```
# For n > 100
analyze_large_dataset <- function(x, sample_size = 100) {
```

```
  if(length(x) > sample_size) {
    warning("Using sampling for large dataset")
    x <- sample(x, sample_size)
  }
  analyze_topology_factors(x)
}

result <- analyze_large_dataset(x)
```

```
## Warning in analyze_large_dataset(x): Using sampling for large dataset
```




```
print(head(result))
```

```
##   factor threshold base_size max_set_size min_set_size
## 1      1    0.6000        90          100            0
## 2      2    0.3000        69          100            0
## 3      4    0.1500        63          100            0
## 4      8    0.0750        33          100            0
## 5     16    0.0375        33          100            0
```

5. Limitations and Considerations

5.1. Computational Limitations

- Functions assume elements are sequentially numbered from 1.
- For very large topologies, computation time may be significant.

- Graph-based methods require more memory than the manual method.

5.2. Approximations and Validity

For large data sets, recommended approaches include:

1. Use `is_topology_connected_manual()` for initial quick evaluation
2. Apply `is_topology_connected2()` for more rigorous analyses when necessary
3. Consider subsampling for extremely large sets.

6. References

- Kelley, J. L. (1955). *General Topology*. Buenos Aires University Press.
- Kolmogórov, A. N., & Fomin, S. V. (1978). *Elements of Function Theory and Functional Analysis*. MIR Publishers.
- Nada, S., El Atik, A. E. F., & Atef, M. (2018). New types of topological structures via graphs. *Mathematical Methods in the Applied Sciences*, 41(15), 5801-5810.
- Trudeau, R. J. (1993). Introduction to Graph Theory. Dover Publications.