Topological Data Analysis in R

Luis Scoccola
lscoccol@uwo.ca

University of Western Ontario

July 19, 2018

Outline

- Brief intro to R
- Persistent homology
- Distance functions
- Other tools

The R language

R is a *programming language* and *environment* for statistical computing and graphics.

As a language, it is an interpreted language. In practice, this means that users access it through a command-line interpreter.

There are several graphical front-ends, such as RStudio and RStudio Server. We will use RStudio.

Vectors

R operates on data structures. The simplest such structure is the numeric **vector**.

- An object representing a vector containing 2.3, 1, and 5 is created by: `c(2.3, 1, 5)`.
- To give the name x to this vector, we use `x <- c(2.3, 1, 5)`.

Vectors support arithmetic operations such as: `x * 3`.

There is a special "number" `NA`, that represents a quantity that is not known.

Indexing vectors

Vectors are indexed starting from 1, and square brackets `[]` are used for indexing.

- Create a vector object: `y <- c(3, NA, -8)`.
- Get its third element: `y[3]`.
- Get the vector consisting of the entries different from *NA* and greater that 0: `y[!is.na(y) & y>0]`.

Here we are using logical operators. Common logical operators include **not** !, **and** &, and **or** |.

Data frames

Data frames are like matrices where the entries can be of different types. A better approximation is to think of data frames as lists (columns) of vectors.

Each list might have a name, and indexing by name is done with `$`.

- Create a data frame object: `experiment <- data.frame(Xvalue=x, Yvalue=y)`.
- Get the x measurements: `experiment$Xvalue`.
- Get the data frame that only contains rows with the y coordinate different from `NA`: `experiment[!is.na(experiment$Yvalue), ]`.

Persistent homology in R

We will use the `TDA` package, roughly following:

Wasserman
Topological Data Analysis.

Fasy, Kim, Lecci, Maria
Introduction to the R package TDA.

Persistent homology

Our first example involves threads that share some resources. Each thread executes an algorithm that involves 5 steps. We run the threads many times in parallel, and every now and then we check which step they are executing and record this.

Go to file `threads.R`...

In this example, homological features represent shared resources.

Persistent homology (cont.)

Next, we study the correlation between temperature and pressure in a fixed city[1].

Go to file `weather.R`...

[1] The data set was obtained from `http://climate.weather.gc.ca/historical_data/search_historic_data_e.html`.

Persistent homology (cont.)

In the previous examples, the Vietoris-Rips complex worked very well.

But in general, this approach is very susceptible to noise and outliers.

Moreover, for very large data sets, computing persistent homology using the VR complex might take very long. So some way of "downsampling" the data is useful.

Distance functions

Definition

Given a subset X of $\mathbb{R}^n$, its **distance function** $d_X : \mathbb{R}^n \to \mathbb{R}$ *is defined by*

$$d_X(y) = \inf_{x \in X} \|y - x\|.$$

Definition

For each $\varepsilon \geq 0$, the **lower level set** *of the distance function is*

$$L_\varepsilon = \{y \in Y \mid d_X(y) < \varepsilon\}.$$

Notice that

$$L_\varepsilon = \bigcup_{x \in X} B(x, \varepsilon).$$

Distance functions (cont.)

When $X = \{x_i\}$ is a point cloud, the homology of each L_ε is isomorphic to the homology of the Čech complex C_ε associated to the covering $\{B(x_i, \varepsilon)\}$.

One can show that the homological features that persist long enough in C are the same as the ones that persist long enough in the Vietoris-Rips complex.

Let's look at our example `weather.R`...

Distance functions (cont.)

Distance functions let us use statistical methods.

For example, the empirical distance function might vary a lot with noise and outliers.

That's why we usually use the data to *infer a distance function*, and then calculate the persistence homology of the inferred distance function.

There are several approaches for this.

Distance functions (cont.)

One possibility is to *smoothen* the original distance function, using the **distance-to-a-measure** (DTM) distance function. This procedure needs a parameter $0 < m_0 < 1$.

Another possibility is to *estimate*, using for example, a **Gaussian Kernel Density Estimator** (KDE), or a **Kernel distance estimator**. Both these methods need a parameter h.

Back to `weather.R`...

Bootstrapping and confidence bands

Another big advantage of using distance functions, is that we can use bootstrapping methods to construct confidence bands.

Bootstrapping is used to assign measures of accuracy to sample estimates.

The main idea is that inference about a distribution from sample data, can be modelled by resampling the sample data and performing inference about the sample from the resampled data.

In our case, we use bootstrapping to construct confidence bands for homological features, as in [2] and [3].

Back to `weather.R`...

Choosing a grid

Distance functions can also be used as a way of "downsampling" large data sets.

This is done by choosing a coarse grid, that has less points than the data set.

We show how this works with a large data set[2], comparing prices of two different assets during lots of consecutive days (`prices.R`).

[2] Data taken from `https://www.alphavantage.co/`.

Other packages

- The `dbscan` package has an implementation of the HDBSCAN algorithm.
- The `TDAmapper` package has an implementation of the Mapper algorithm.

In the R files you can find some very simple examples on how to use these packages.

What about Python?

The Dionysus library is actually implemented in C++, and it has a wrapper for Python too. See the documentation at:
`http://www.mrzv.org/software/dionysus2/`

The TDA wrapper for Python has access to lower level functions. This makes it a bit more involved to use, but it also gives more flexibility.

Both Mapper and HDBSCAN also exist for Python. See

`https://github.com/scikit-learn-contrib/hdbscan`
`http://danifold.net/mapper/`

Some takeaways

- Try to plot your data, or some simplification of it, before applying any fancy technique to it.
- Be careful with the scales of the different dimensions of your data, when applying topological tools.
- When using VR, start with a small max radius, since the algorithm's time complexity is exponential with respect to the max radius.
- If possible, use confidence bands to differentiate real features from noise.
- Different libraries perform differently, so try another one if the processing is taking too long. In my experience, Dionysus is not the fastest, but it is very stable.

More refereces I


W. N. Venables, D. M. Smith and the R Core Team
An Introduction to R.


Chazal, Fasy, Lecci, Michel, Rinaldo, Wasserman
Robust Topological Inference: Distance To a Measure and Kernel Distance


Fasy, Lecci, Rinaldo, Wasserman, Balakrishnan, Singh
Confidence sets for persistence diagrams