An introduction to persistent homology

Part 3: Computation

Ulrich Bauer

TUM

Aug 7, 2019

Summer School on Persistent Homology and Barcodes
Schloss Rauischholzhausen

Vietoris–Rips complexes

Consider a finite metric space (X, d).
The *Vietoris–Rips complex* is the simplicial complex

$$\mathrm{Rips}_t(X) = \{S \subseteq X \mid \operatorname{diam} S \leq t\}$$

- all edges with pairwise distance $\leq t$
- all possible higher simplices

Vietoris–Rips complexes

Consider a finite metric space (X, d).
The *Vietoris–Rips complex* is the simplicial complex

$$\mathrm{Rips}_t(X) = \{S \subseteq X \mid \operatorname{diam} S \leq t\}$$

- all edges with pairwise distance $\leq t$
- all possible higher simplices

For large t, $\mathrm{Rips}_t(X)$ is the full simplex with vertices X

- Number of d-simplices is $\binom{|X|}{d+1}$
- Computation is one of the most important challenges in applied topology!

An example computation

Example data set:

- 192 points on $\mathbb{S}^2$
- persistent homology barcodes up to dimension 2
- over 56 mio. simplices in 3-skeleton

An example computation

Example data set:

- 192 points on $\mathbb{S}^2$
- persistent homology barcodes up to dimension 2
- over 56 mio. simplices in 3-skeleton

Some previous software:

- javaplex (Stanford): 3200 seconds, 12 GB
- Dionysus (Duke): 615 seconds, 3.4 GB
- DIPHA (IST Austria): 50 seconds, 6 GB
- GUDHI (INRIA): 60 seconds, 3 GB

An example computation

Example data set:

- 192 points on $\mathbb{S}^2$
- persistent homology barcodes up to dimension 2
- over 56 mio. simplices in 3-skeleton

Some previous software:

- javaplex (Stanford): 3200 seconds, 12 GB
- Dionysus (Duke): 615 seconds, 3.4 GB
- DIPHA (IST Austria): 50 seconds, 6 GB
- GUDHI (INRIA): 60 seconds, 3 GB

Demo: live.ripser.org

An example computation

Example data set:

- 192 points on $\mathbb{S}^2$
- persistent homology barcodes up to dimension 2
- over 56 mio. simplices in 3-skeleton

Some previous software:

- javaplex (Stanford): 3200 seconds, 12 GB
- Dionysus (Duke): 615 seconds, 3.4 GB
- DIPHA (IST Austria): 50 seconds, 6 GB
- GUDHI (INRIA): 60 seconds, 3 GB

Demo: live.ripser.org

- Ripser: 1.2 seconds, 160 MB

Ripser

A software for computing Vietoris–Rips persistence barcodes

- around 1000 lines of C++ code, no external dependencies
- support for
  - coefficients in a prime field $\mathbb{Z}/p\mathbb{Z}$
  - sparse distance matrices (for distance threshold)
- open source (`http://ripser.org`)
- online version (`http://live.ripser.org`)
- 2016 ATMCS Best New Software Award (joint with RIVET by M. Lesnick and M. Wright)

The four special ingredients

Main improvements to the standard algorithm:

- Clearing inessential columns [Chen, Kerber 2011]
- Computing cohomology [de Silva et al. 2011]
- Implicit matrix reduction
- Apparent and emergent pairs

The four special ingredients

Main improvements to the standard algorithm:

- Clearing inessential columns [Chen, Kerber 2011]
- Computing cohomology [de Silva et al. 2011]
- Implicit matrix reduction
- Apparent and emergent pairs

Lessons from previous work:

- Clearing and cohomology yield considerable speedup,
- but only when *both* are used in conjuction!

Filtrations and refinements

Simplexwise filtrations

Call a filtration $K_\bullet = (K_i)_{i \in I}$ of simplicial complexes (I totally ordered)

- *essential* if $i \neq j$ implies $K_i \neq K_j$,
- *simplexwise* if all $K_i \neq \varnothing$ are of the form $K_i = K_j \cup \{\sigma_i\}$ for some $j < i \in I$ and $\sigma_i \in K$.

Simplexwise filtrations

Call a filtration $K_\bullet = (K_i)_{i \in I}$ of simplicial complexes (I totally ordered)

- *essential* if $i \neq j$ implies $K_i \neq K_j$,
- *simplexwise* if all $K_i \neq \varnothing$ are of the form $K_i = K_j \cup \{\sigma_i\}$ for some $j < i \in I$ and $\sigma_i \in K$.

Note:

- These properties are natural assumptions for computation.
- The *Vietoris–Rips filtration* (indexed by $\mathbb{R}$) is not simplexwise (and not essential).
- To compute Vietoris–Rips persistence, we have to reindex and refine.

Reindexing and refinement

A filtration $K : I \to \mathbf{Simp}$ is a *reindexing* of another filtration $F : R \to \mathbf{Simp}$ if $F = K \circ r$ for some monotonic map $r : R \to I$.

- If r is injective, we call K a *refinement* of F;
- if r is surjective, we call K a *condensation* of F.

Reindexing and refinement

A filtration $K : I \to \mathbf{Simp}$ is a *reindexing* of another filtration $F : R \to \mathbf{Simp}$ if $F = K \circ r$ for some monotonic map $r : R \to I$.

- If r is injective, we call K a *refinement* of F;
- if r is surjective, we call K a *condensation* of F.

The Vietoris–Rips filtration can be reindexed:

- condensed to an essential filtration (over the finite set of pairwise distances), and further
- refined to a simplexwise filtration.

Reindexing and refinement

A filtration $K : I \to \mathbf{Simp}$ is a *reindexing* of another filtration $F : R \to \mathbf{Simp}$ if $F = K \circ r$ for some monotonic map $r : R \to I$.

- If r is injective, we call K a *refinement* of F;
- if r is surjective, we call K a *condensation* of F.

The Vietoris–Rips filtration can be reindexed:

- condensed to an essential filtration (over the finite set of pairwise distances), and further
- refined to a simplexwise filtration.

Ripser use the *lexicographic refinement*: simplices ordered by

- diameter, then
- dimension, then
- (decreasing) lexicographic order of vertices $\{v_{i_k}, \dots, v_{i_0}\}$ (induced by fixed total order $v_0 < \cdots < v_{n-1}$ on vertices)

Enumerating (co)faces in lexicographic order

There is an order-preserving bijection

$$\begin{array}{ccccccc} [v_2, v_1, v_0] & [v_3, v_1, v_0] & [v_3, v_1, v_0] & [v_3, v_2, v_0] & \cdots & [v_{n-3}, v_{n-2}, v_{n-1}] \\ \downarrow & \downarrow & \downarrow & \downarrow & & \downarrow \\ 0 & 1 & 2 & 3 & \cdots & \binom{n}{k+1} - 1 \end{array}$$

from k-simplices (as ordered tuples, ordered lexicographically) to natural numbers (called *combinatorial number system*).

- One direction is given by

$$[v_{i_k}, \ldots, v_{i_0}] \mapsto \sum_{j=0}^{k} \binom{i_j}{j+1}$$

- The other direction can also be computed quickly
- Using this bijection, faces and cofaces can be efficiently enumerated in (decreasing) lexicographic order

Persistence barcodes through reindexing

Proposition

Consider a filtration $F : \mathbb{R} \to \mathbf{Simp}$ *with reindexing* $F = K \circ r$, $K : I \to \mathbf{Simp}$, $r : \mathbb{R} \to I$.
The persistence barcode of $K_\bullet$ *determines the persistence barcode of* $F_\bullet$*:*

$$B(H_*(F_\bullet)) = \{r^{-1}[i,j) \neq \varnothing \mid [i,j) \in B(H_*(K_\bullet))\}.$$

Matrix reduction

Computing homology

Computing homology $H_* = Z_*/B_*$:

- compute basis for boundaries $B_* = \operatorname{im} \partial_*$
- extend to basis for cycles $Z_* = \ker \partial_*$
- new (non-boundary) basis cycles generate quotient Z_*/B_*

Computing homology

Computing homology $H_* = Z_*/B_*$:

- compute basis for boundaries $B_* = \operatorname{im} \partial_*$
- extend to basis for cycles $Z_* = \ker \partial_*$
- new (non-boundary) basis cycles generate quotient Z_*/B_*

Computing *persistent* homology $H_* = Z_*/B_*$ (for a simplexwise filtration $K_i \subseteq K$):

- compute *filtered* basis for boundaries $B_* = \operatorname{im} \partial_*$
- extend to basis for cycles $Z_* = \ker \partial_*$
- all basis cycles generate *persistent* homology

Persistence by matrix reduction

Given:

- D: matrix of boundary ∂_d for a simplexwise filtration $(K_i)_i$ (for canonical basis in filtration order, indexed by $I \times J$)

Wanted:

- persistence barcode of homology $H_d(K_i; \mathbb{F})$ for some (prime) field $\mathbb{F}$, in dimensions $d = 0, \ldots, k$

Notation:

- M_j: column j of M; m_{ij}: entry in row i and column j
- $\operatorname{PivotIndex} M_j = \min\{i \in I : m_{kj} = 0 \text{ for all } k > i\}$.

Persistence by matrix reduction

Given:

- D: matrix of boundary ∂_d for a simplexwise filtration $(K_i)_i$ (for canonical basis in filtration order, indexed by $I \times J$)

Wanted:

- persistence barcode of homology $H_d(K_i; \mathbb{F})$ for some (prime) field $\mathbb{F}$, in dimensions $d = 0, \ldots, k$

Notation:

- M_j: column j of M; m_{ij}: entry in row i and column j
- $\operatorname{PivotIndex} M_j = \min\{i \in I : m_{kj} = 0 \text{ for all } k > i\}$.

Computation: barcode is obtained by *matrix reduction* of D

- $R = D \cdot V$ *reduced* (non-zero columns have distinct pivots)
- V is regular upper triangular

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$$I_b = \{i : R_i = 0\} \qquad \textit{birth} \text{ indices ,}$$
$$I_d = \{j : R_j \neq 0\} \qquad \textit{death} \text{ indices,}$$
$$I_e = I_b \setminus \operatorname{PivotIndices} R \qquad \textit{essential} \text{ indices.}$$

(note: $i = \operatorname{PivotIndex} R_j$ implies $R_i = 0$, thus $\operatorname{PivotIndices} R \subseteq I_b$). Then

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$$I_b = \{i : R_i = 0\} \qquad \textit{birth} \text{ indices ,}$$
$$I_d = \{j : R_j \neq 0\} \qquad \textit{death} \text{ indices,}$$
$$I_e = I_b \setminus \operatorname{PivotIndices} R \qquad \textit{essential} \text{ indices.}$$

(note: $i = \operatorname{PivotIndex} R_j$ implies $R_i = 0$, thus $\operatorname{PivotIndices} R \subseteq I_b$). Then

$$\Sigma_B = \{R_j \mid j \in I_d\} \qquad \text{is a basis of } B_*,$$

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$$I_b = \{i : R_i = 0\} \qquad \textit{birth} \text{ indices},$$
$$I_d = \{j : R_j \neq 0\} \qquad \textit{death} \text{ indices},$$
$$I_e = I_b \setminus \operatorname{PivotIndices} R \qquad \textit{essential} \text{ indices}.$$

(note: $i = \operatorname{PivotIndex} R_j$ implies $R_i = 0$, thus $\operatorname{PivotIndices} R \subseteq I_b$). Then

$$\Sigma_B = \{R_j \mid j \in I_d\} \qquad \text{is a basis of } B_*,$$
$$\Sigma_Z = \Sigma_B \cup \{R_i \mid i \in I_e\} \qquad \text{is a } \textit{compatible} \text{ basis of } Z_*.$$

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$$I_b = \{i : R_i = 0\} \qquad \textit{birth} \text{ indices},$$
$$I_d = \{j : R_j \neq 0\} \qquad \textit{death} \text{ indices},$$
$$I_e = I_b \setminus \operatorname{PivotIndices} R \qquad \textit{essential} \text{ indices}.$$

(note: $i = \operatorname{PivotIndex} R_j$ implies $R_i = 0$, thus $\operatorname{PivotIndices} R \subseteq I_b$). Then

$$\Sigma_B = \{R_j \mid j \in I_d\} \qquad \text{is a basis of } B_*,$$
$$\Sigma_Z = \Sigma_B \cup \{R_i \mid i \in I_e\} \qquad \text{is a } \textit{compatible} \text{ basis of } Z_*.$$

- Persistent homology is generated by the basis cycles Σ_Z.

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$$I_b = \{i : R_i = 0\} \qquad \textit{birth} \text{ indices},$$
$$I_d = \{j : R_j \neq 0\} \qquad \textit{death} \text{ indices},$$
$$I_e = I_b \setminus \operatorname{PivotIndices} R \qquad \textit{essential} \text{ indices}.$$

(note: $i = \operatorname{PivotIndex} R_j$ implies $R_i = 0$, thus $\operatorname{PivotIndices} R \subseteq I_b$). Then

$$\Sigma_B = \{R_j \mid j \in I_d\} \qquad \text{is a basis of } B_*,$$
$$\Sigma_Z = \Sigma_B \cup \{R_i \mid i \in I_e\} \qquad \text{is a } \textit{compatible} \text{ basis of } Z_*.$$

- Persistent homology is generated by the basis cycles Σ_Z.
- Persistence intervals: $\{[i, j) \mid i = \operatorname{PivotIndex} R_j\} \cup \{[i, \infty) \mid i \in I_e\}$

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

R

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1				
4				1			
5					1		
6						1	
7							1

V

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

R

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1				
4				1			
5					1		
6						1	
7							1

V

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

R

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1				
4				1			
5					1		
6						1	
7							1

V

Matrix reduction

	1	2	3	4	5	6	7
1			1		1	1	
2			1			1	
3							1
4					1	0	
5							1
6							1
7							

R

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1				
4				1			
5					1	1	
6						1	
7							1

V

Matrix reduction

	1	2	3	4	5	6	7
1			1		1	1	
2			1			1	
3							1
4					1		
5							1
6							1
7							

R

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1				
4				1			
5					1	1	
6						1	
7							1

V

Matrix reduction

	1	2	3	4	5	6	7
1			1		1	0	
2			1			0	
3							1
4					1		
5							1
6							1
7							

R

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

V

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

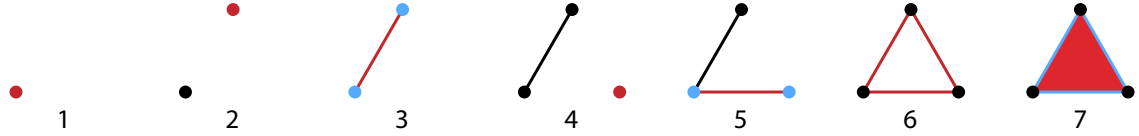
R

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

V

Matrix reduction



	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

R

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

V

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

R

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

V

Matrix reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

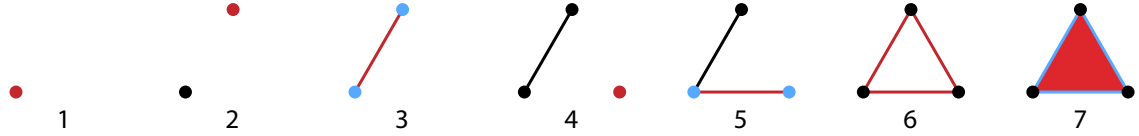
R

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

V

Matrix reduction



	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

R

$= D \cdot$

	1	2	3	4	5	6	7
1	1						
2		1					
3			1			1	
4				1			
5					1	1	
6						1	
7							1

V

Matrix reduction algorithm

Require: D: $I \times J$ matrix

Ensure: $R = D \cdot V$: reduced, V: regular upper triangular, P: persistence pairs, E: essential

$P := \varnothing$

$R := D$

$V := I$

for $j \in J$ in increasing order **do**

  while $\exists k < j$ with $\operatorname{PivotIndex} R_k = \operatorname{PivotIndex} R_j$ **do**

    $\lambda := \operatorname{PivotEntry} R_j / \operatorname{PivotEntry} R_k$

    $R_j := R_j - \lambda \cdot R_k$ ▷ elimintates $\operatorname{PivotEntry} R_j$

    $V_j := V_j - \lambda \cdot V_k$

  if $(i := \operatorname{PivotIndex} R_j) \neq 0$ **then**

    append (i, j) to P ▷ persistence interval $[i, j)$

  else

    append j to E ▷ persistence interval $[j, \infty)$

Clearing

Clearing non-essential positive columns

Idea [Chen, Kerber 2011]:

- Don't reduce at non-essential birth indices
- Use the fact that $i = \operatorname{PivotIndex} R_j$ implies $R_i = 0$
  - Reduce matrices of $\partial_d : C_d \to C_{d-1}$ for $d = k+1, \dots, 1$
  - Whenever $i = \operatorname{PivotIndex} R_j$ (in matrix for ∂_d): set R_i to 0 (in matrix for ∂_{d-1})

Clearing non-essential positive columns

Idea [Chen, Kerber 2011]:

- Don't reduce at non-essential birth indices
- Use the fact that $i = \operatorname{PivotIndex} R_j$ implies $R_i = 0$
  - Reduce matrices of $\partial_d : C_d \to C_{d-1}$ for $d = k+1, \dots, 1$
  - Whenever $i = \operatorname{PivotIndex} R_j$ (in matrix for ∂_d): set R_i to 0 (in matrix for ∂_{d-1})

Note:

- reducing *birth* columns typically harder than death columns:
  - $O(j^2)$ (birth) vs. $O((j-i)^2)$ (death)
- with clearing: need only reduce *essential* columns

Counting homology column reductions

Consider $k+1$-skeleton of $n-1$-simplex (Rips filtration)

Number of columns in boundary matrix:

$$\underbrace{\sum_{d=1}^{k+1}\binom{n}{d+1}}_{\text{total}} = \underbrace{\sum_{d=1}^{k+1}\binom{n-1}{d}}_{\text{death}} + \underbrace{\sum_{d=1}^{k+1}\binom{n-1}{d+1}}_{\text{birth}}$$

Counting homology column reductions

Consider $k+1$-skeleton of $n-1$-simplex (Rips filtration)

Number of columns in boundary matrix:

$$\underbrace{\sum_{d=1}^{k+1}\binom{n}{d+1}}_{\text{total}} = \underbrace{\sum_{d=1}^{k+1}\binom{n-1}{d}}_{\text{death}} + \underbrace{\sum_{d=1}^{k+1}\binom{n-1}{d+1}}_{\text{birth}}$$

$$= \underbrace{\sum_{d=1}^{k+1}\binom{n-1}{d}}_{\text{death}} + \underbrace{\binom{n-1}{k+2}}_{\text{essential}} + \underbrace{\sum_{d=1}^{k}\binom{n-1}{d+1}}_{\text{cleared}}$$

Counting homology column reductions

Consider $k+1$-skeleton of $n-1$-simplex (Rips filtration)

Number of columns in boundary matrix:

$$\underbrace{\sum_{d=1}^{k+1}\binom{n}{d+1}}_{\text{total}} = \underbrace{\sum_{d=1}^{k+1}\binom{n-1}{d}}_{\text{death}} + \underbrace{\sum_{d=1}^{k+1}\binom{n-1}{d+1}}_{\text{birth}}$$

$$= \underbrace{\sum_{d=1}^{k+1}\binom{n-1}{d}}_{\text{death}} + \underbrace{\binom{n-1}{k+2}}_{\text{essential}} + \underbrace{\sum_{d=1}^{k}\binom{n-1}{d+1}}_{\text{cleared}}$$

$k=2, n=192$: $\quad 56\,050\,096 = 1\,161\,471 + 53\,727\,345 + 1\,161\,280$

- Clearing didn't help much!

Cohomology

Persistent (co)homology

How is persistent cohomology related to persistent homology?

Proposition (de Silva et al 2011)

Persistent homology and persistent cohomology have the same barcode.

Persistent (co)homology

How is persistent cohomology related to persistent homology?

Proposition (de Silva et al 2011)

Persistent homology and persistent cohomology have the same barcode.

Proof.

- Cohomology (over $\mathbb{F}$) is vector space dual to homology:

$$H^p(K) \cong H_p(K)^* = \mathrm{Hom}(H_p(K), \mathbb{F}).$$

Persistent (co)homology

How is persistent cohomology related to persistent homology?

Proposition (de Silva et al 2011)

Persistent homology and persistent cohomology have the same barcode.

Proof.

- Cohomology (over $\mathbb{F}$) is vector space dual to homology:

$$H^p(K) \cong H_p(K)^* = \mathrm{Hom}(H_p(K), \mathbb{F}).$$

- Duality preserves ranks of linear maps (in finite dimensions): for $f : V \to W$ with dual $f^* : W^* \to V^*$,

$$\mathrm{rank}\, f^* = \mathrm{rank}\, f.$$

Persistent (co)homology

How is persistent cohomology related to persistent homology?

Proposition (de Silva et al 2011)

Persistent homology and persistent cohomology have the same barcode.

Proof.

- Cohomology (over $\mathbb{F}$) is vector space dual to homology:

$$H^p(K) \cong H_p(K)^* = \mathrm{Hom}(H_p(K), \mathbb{F}).$$

- Duality preserves ranks of linear maps (in finite dimensions): for $f : V \to W$ with dual $f^* : W^* \to V^*$,

$$\mathrm{rank}\, f^* = \mathrm{rank}\, f.$$

- The persistence barcode of a persistence module is uniquely determined by the ranks of the internal maps. □

Persistent (relative) homology

How is persistent relative homology related to persistent homology?

- Consider the short exact sequence of filtered chain complexes (with coefficients in $\mathbb{F}$)

$$0 \to C_p(K_\bullet) \to C_p(K) \to C_p(K, K_\bullet) \to 0$$

  (where $C_p(K)$ is the constant filtration).

- This induces a long exact sequence of persistence modules

$$\cdots \to H_{p+1}(K, K_\bullet) \to H_p(K_\bullet) \to H_p(K) \to H_p(K, K_\bullet) \to \cdots$$

  and analogously for persistent cohomology.

Persistent (relative) homology barcodes

Consider the long exact sequence of persistence modules

$$\cdots \to H_{p+1}(K) \xrightarrow{r} H_{p+1}(K, K_\bullet) \xrightarrow{\partial} H_p(K_\bullet) \xrightarrow{i} H_p(K) \to \cdots$$

We have:

- $B(\mathrm{im}(i)) = B^\infty(H_p(K_\bullet)) := \{[b, \infty) \in B(H_p(K_\bullet))\}$

Persistent (relative) homology barcodes

Consider the long exact sequence of persistence modules

$$\cdots \to H_{p+1}(K) \xrightarrow{r} H_{p+1}(K, K_\bullet) \xrightarrow{\partial} H_p(K_\bullet) \xrightarrow{i} H_p(K) \to \cdots$$

We have:

- $B(\mathrm{im}(i)) = B^\infty(H_p(K_\bullet)) := \{[b, \infty) \in B(H_p(K_\bullet))\}$
- $B(\mathrm{im}(\partial)) = B(\ker(i)) = B^\dagger(H_p(K_\bullet)) := \{[b, d) \in B(H_p(K_\bullet)) \mid d < \infty\}$

Persistent (relative) homology barcodes

Consider the long exact sequence of persistence modules

$$\cdots \to H_{p+1}(K) \xrightarrow{r} H_{p+1}(K, K_\bullet) \xrightarrow{\partial} H_p(K_\bullet) \xrightarrow{i} H_p(K) \to \cdots$$

We have:

- $B(\mathrm{im}(i)) = B^\infty(H_p(K_\bullet)) := \{[b, \infty) \in B(H_p(K_\bullet))\}$
- $B(\mathrm{im}(\partial)) = B(\ker(i)) = B^\dagger(H_p(K_\bullet)) := \{[b, d) \in B(H_p(K_\bullet)) \mid d < \infty\}$

The sequence splits at $H_p(K_\bullet) \cong \mathrm{im}(\partial) \oplus \mathrm{im}(i)$.

Persistent (relative) homology barcodes

Consider the long exact sequence of persistence modules

$$\cdots \to H_{p+1}(K) \xrightarrow{r} H_{p+1}(K, K_\bullet) \xrightarrow{\partial} H_p(K_\bullet) \xrightarrow{i} H_p(K) \to \cdots$$

We have:

- $B(\mathrm{im}(i)) = B^\infty(H_p(K_\bullet)) := \{[b, \infty) \in B(H_p(K_\bullet))\}$
- $B(\mathrm{im}(\partial)) = B(\ker(i)) = B^\dagger(H_p(K_\bullet)) := \{[b, d) \in B(H_p(K_\bullet)) \mid d < \infty\}$

The sequence splits at $H_p(K_\bullet) \cong \mathrm{im}(\partial) \oplus \mathrm{im}(i)$.

- $B(\ker(\partial)) = B(\mathrm{im}(r)) = \overline{B^\infty}(H_p(K_\bullet)) := \{(-\infty, b) \mid [b, \infty) \in B(H_p(K_\bullet))\}$

Persistent (relative) homology barcodes

Consider the long exact sequence of persistence modules

$$\cdots \to H_{p+1}(K) \xrightarrow{r} H_{p+1}(K, K_\bullet) \xrightarrow{\partial} H_p(K_\bullet) \xrightarrow{i} H_p(K) \to \cdots$$

We have:

- $B(\mathrm{im}(i)) = B^\infty(H_p(K_\bullet)) := \{[b, \infty) \in B(H_p(K_\bullet))\}$
- $B(\mathrm{im}(\partial)) = B(\ker(i)) = B^\dagger(H_p(K_\bullet)) := \{[b, d) \in B(H_p(K_\bullet)) \mid d < \infty\}$

The sequence splits at $H_p(K_\bullet) \cong \mathrm{im}(\partial) \oplus \mathrm{im}(i)$.

- $B(\ker(\partial)) = B(\mathrm{im}(r)) = \overline{B^\infty}(H_p(K_\bullet)) := \{(-\infty, b) \mid [b, \infty) \in B(H_p(K_\bullet))\}$

The sequence splits at $H_{p+1}(K, K_\bullet) \cong \mathrm{im}(\partial) \oplus \mathrm{im}(r)$.

Proposition (de Silva et al. 2011)

The absolute *and* relative *persistence barcodes determine each other:*

- $B^\dagger(H_p(K_\bullet)) = B^\dagger(H_{p+1}(K, K_\bullet))$

Persistent (relative) homology barcodes

Consider the long exact sequence of persistence modules

$$\cdots \to H_{p+1}(K) \xrightarrow{r} H_{p+1}(K, K_\bullet) \xrightarrow{\partial} H_p(K_\bullet) \xrightarrow{i} H_p(K) \to \cdots$$

We have:

- $B(\mathrm{im}(i)) = B^\infty(H_p(K_\bullet)) := \{[b, \infty) \in B(H_p(K_\bullet))\}$
- $B(\mathrm{im}(\partial)) = B(\ker(i)) = B^\dagger(H_p(K_\bullet)) := \{[b, d) \in B(H_p(K_\bullet)) \mid d < \infty\}$

The sequence splits at $H_p(K_\bullet) \cong \mathrm{im}(\partial) \oplus \mathrm{im}(i)$.

- $B(\ker(\partial)) = B(\mathrm{im}(r)) = \overline{B^\infty}(H_p(K_\bullet)) := \{(-\infty, b) \mid [b, \infty) \in B(H_p(K_\bullet))\}$

The sequence splits at $H_{p+1}(K, K_\bullet) \cong \mathrm{im}(\partial) \oplus \mathrm{im}(r)$.

Proposition (de Silva et al. 2011)

The absolute *and* relative *persistence barcodes determine each other:*

- $B^\dagger(H_p(K_\bullet)) = B^\dagger(H_{p+1}(K, K_\bullet))$
- $B^\infty(H_p(K_\bullet)) = \overline{B^\infty}(H_p(K, K_\bullet))$

Cohomology and clearing

- Cohomology allows for clearing starting from dimension 0

Cohomology and clearing

- Cohomology allows for clearing starting from dimension 0
- Persistence in dimension 0 has special algorithms
  - Kruskal's algorithm for minimum spanning tree
  - union-find data structure

Cohomology and clearing

- Cohomology allows for clearing starting from dimension 0
- Persistence in dimension 0 has special algorithms
  - Kruskal's algorithm for minimum spanning tree
  - union-find data structure
- Persistent cohomology arises from a *reverse* filtration

$$C^*(K_0) \twoheadleftarrow C^*(K_1) \twoheadleftarrow \cdots \twoheadleftarrow C^*(K_n)$$

Cohomology and clearing

- Cohomology allows for clearing starting from dimension 0
- Persistence in dimension 0 has special algorithms
  - Kruskal's algorithm for minimum spanning tree
  - union-find data structure
- Persistent cohomology arises from a *reverse* filtration

$$C^*(K_0) \twoheadleftarrow C^*(K_1) \twoheadleftarrow \cdots \twoheadleftarrow C^*(K_n)$$

- Persistent *relative* cohomology arises from a filtration

$$C^*(K, K_0) \hookleftarrow C^*(K, K_1) \hookleftarrow \cdots \hookleftarrow C^*(K, K_n)$$

Cohomology and clearing

- Cohomology allows for clearing starting from dimension 0
- Persistence in dimension 0 has special algorithms
  - Kruskal's algorithm for minimum spanning tree
  - union-find data structure
- Persistent cohomology arises from a *reverse* filtration

$$C^*(K_0) \twoheadleftarrow C^*(K_1) \twoheadleftarrow \cdots \twoheadleftarrow C^*(K_n)$$

- Persistent *relative* cohomology arises from a filtration

$$C^*(K, K_0) \hookleftarrow C^*(K, K_1) \hookleftarrow \cdots \hookleftarrow C^*(K, K_n)$$

  - Can be computed by reduction of *filtration coboundary matrix*:
    - transpose of filtration boundary matrix,
    - with reversed index orders

Counting cohomology column reductions

Consider K: $k+1$-skeleton of $n-1$-simplex, Rips filtration

- Number of columns in coboundary matrix:

$$\underbrace{\sum_{d=0}^{k} \binom{n}{d+1}}_{\text{total}} = \underbrace{\sum_{d=0}^{k} \binom{n-1}{d+1}}_{\text{death}} + \underbrace{\sum_{d=0}^{k} \binom{n-1}{d}}_{\text{birth}}$$

Counting cohomology column reductions

Consider K: $k+1$-skeleton of $n-1$-simplex, Rips filtration

- Number of columns in coboundary matrix:

$$\underbrace{\sum_{d=0}^{k}\binom{n}{d+1}}_{\text{total}} = \underbrace{\sum_{d=0}^{k}\binom{n-1}{d+1}}_{\text{death}} + \underbrace{\sum_{d=0}^{k}\binom{n-1}{d}}_{\text{birth}}$$

$$= \underbrace{\sum_{d=0}^{k}\binom{n-1}{d+1}}_{\text{death}} + \underbrace{\binom{n-1}{0}}_{\text{essential}} + \underbrace{\sum_{d=1}^{k}\binom{n-1}{d}}_{\text{cleared}}$$

Counting cohomology column reductions

Consider K: $k+1$-skeleton of $n-1$-simplex, Rips filtration

- Number of columns in coboundary matrix:

$$\underbrace{\sum_{d=0}^{k}\binom{n}{d+1}}_{\text{total}} = \underbrace{\sum_{d=0}^{k}\binom{n-1}{d+1}}_{\text{death}} + \underbrace{\sum_{d=0}^{k}\binom{n-1}{d}}_{\text{birth}}$$

$$= \underbrace{\sum_{d=0}^{k}\binom{n-1}{d+1}}_{\text{death}} + \underbrace{\binom{n-1}{0}}_{\text{essential}} + \underbrace{\sum_{d=1}^{k}\binom{n-1}{d}}_{\text{cleared}}$$

- $k=2, n=192$: $1\,179\,808 = 1\,161\,471 + 1 + 18\,336$

Observations

For a typical input:

- V has very few off-diagonal entries
- most death columns of D are already reduced

Observations

For a typical input:

- V has very few off-diagonal entries
- most death columns of D are already reduced

Previous example ($k = 2, n = 192$):

- Only $79 + 42 + 1 = 122$ out of $192 + 18\,145 + 1\,143\,135 = 1\,161\,471$ columns are actually modified in matrix reduction

Implicit matrix reduction

Implicit matrix reduction

Standard approach:

- Boundary matrix D for filtration-ordered basis
  - Explicitly generated and stored in memory

Implicit matrix reduction

Standard approach:

- Boundary matrix D for filtration-ordered basis
  - Explicitly generated and stored in memory
- Matrix reduction: store only reduced matrix R
  - Initialize matrix as D, reduce to R by column operations

Implicit matrix reduction

Standard approach:

- Boundary matrix D for filtration-ordered basis
  - Explicitly generated and stored in memory
- Matrix reduction: store only reduced matrix R
  - Initialize matrix as D, reduce to R by column operations

Approach for Ripser:

- Boundary matrix D for lexicographically ordered basis
  - Columns recomputed on the fly

Implicit matrix reduction

Standard approach:

- Boundary matrix D for filtration-ordered basis
  - Explicitly generated and stored in memory
- Matrix reduction: store only reduced matrix R
  - Initialize matrix as D, reduce to R by column operations

Approach for Ripser:

- Boundary matrix D for lexicographically ordered basis
  - Columns recomputed on the fly
- Matrix reduction: store only coefficient matrix V
  - Columns of $R_j = D \cdot V_j$ recomputed on the fly
  - Typically, V is much sparser and smaller than R

Implicit boundary matrix reduction algorithm

Require: D: $I \times J$ matrix

Ensure: $R = D \cdot V$: reduced, V: regular upper triangular, P: persistence pairs, E: essential

$P := \varnothing$

$R := D$

$V := I$

for $j \in J$ in increasing order **do**

  while $\exists k < j$ with $\operatorname{PivotIndex} R_k = \operatorname{PivotIndex} R_j$ **do**

    $\lambda := \operatorname{PivotEntry} R_j / \operatorname{PivotEntry} R_k$

    $R_j := R_j - \lambda \cdot R_k$ ▷ elimintates $\operatorname{PivotEntry} R_j$

    $V_j := V_j - \lambda \cdot V_k$

  if $(i := \operatorname{PivotIndex} R_j) \neq 0$ **then**

    append (i, j) to P ▷ persistence interval $[i, j)$

  else

    append j to E ▷ persistence interval $[j, \infty)$

Implicit boundary matrix reduction algorithm

Require: D: $I \times J$ matrix

Ensure: $R = D \cdot V$: reduced, V: regular upper triangular, P: persistence pairs, E: essential

$P := \varnothing$

$R := D$

$V := I$

for $j \in J$ in increasing order **do**

  while $\exists k < j$ with $\operatorname{PivotIndex} R_k = \operatorname{PivotIndex} R_j$ **do**

    $\lambda := \operatorname{PivotEntry} R_j / \operatorname{PivotEntry} R_k$

    $R_j := R_j - \lambda \cdot R_k$ ▷ elimintates $\operatorname{PivotEntry} R_j$

    $V_j := V_j - \lambda \cdot V_k$

  if $(i := \operatorname{PivotIndex} R_j) \neq 0$ **then**

    append (i, j) to P ▷ persistence interval $[i, j)$

  else

    append j to E ▷ persistence interval $[j, \infty)$

Implicit boundary matrix reduction algorithm

Require: D: $I \times J$ matrix

Ensure: $R = D \cdot V$: reduced, V: regular upper triangular, P: persistence pairs, E: essential

$P := \varnothing$

$R := D$

$V := I$

for $j \in J$ in increasing order **do**

  while $\exists k < j$ with $\operatorname{PivotIndex} R_k = \operatorname{PivotIndex} R_j$ **do**

    $\lambda := \operatorname{PivotEntry} R_j / \operatorname{PivotEntry} R_k$

    $R_j := R_j - \lambda \cdot D \cdot V_k$ ▷ elimintates $\operatorname{PivotEntry} R_j$

    $V_j := V_j - \lambda \cdot V_k$

  if $(i := \operatorname{PivotIndex} R_j) \neq 0$ **then**

    append (i, j) to P ▷ persistence interval $[i, j)$

  else

    append j to E ▷ persistence interval $[j, \infty)$

Apparent and emergent pairs

Apparent pairs

Definition

In a simplexwise filtration, a pair of simplices (σ, τ) is an *apparent* pair if

- σ is the youngest facet of τ, and
- τ is the oldest cofacet of σ.

Apparent pairs

Definition

In a simplexwise filtration, a pair of simplices (σ, τ) is an *apparent* pair if

- σ is the youngest facet of τ, and
- τ is the oldest cofacet of σ.

Proposition

The apparent pairs are persistence pairs.

Apparent pairs

Definition

In a simplexwise filtration, a pair of simplices (σ, τ) is an *apparent* pair if

- σ is the youngest facet of τ, and
- τ is the oldest cofacet of σ.

Proposition

The apparent pairs are persistence pairs.

Remark

The apparent pairs also form a *discrete gradient* (in the sense of discrete Morse theory).

Emergent persistent pairs

A persistence pair (σ, τ) for a simplexwise filtration is

- an *emergent facet pair* if σ is the youngest facet of τ,
- an *emergent cofacet pair* if τ is the oldest cofacet of σ.

Emergent persistent pairs

A persistence pair (σ, τ) for a simplexwise filtration is

- an *emergent facet pair* if σ is the youngest facet of τ,
- an *emergent cofacet pair* if τ is the oldest cofacet of σ.

Clearly, every apparent pair is both.

Emergent persistent pairs

A persistence pair (σ, τ) for a simplexwise filtration is

- an *emergent facet pair* if σ is the youngest facet of τ,
- an *emergent cofacet pair* if τ is the oldest cofacet of σ.

Clearly, every apparent pair is both.

Proposition

Let $K_\bullet$ be the lexicographically refined Rips filtration. Assume that

- *τ is the lexicographically minimal cofacet of σ with $\operatorname{diam}(\tau) = \operatorname{diam}(\sigma)$, and*
- *τ is not already in a persistence pair (ρ, τ) of $K_\bullet$ with $\rho \succ \sigma$.*

Then (σ, τ) is a 0-persistence emergent coface pair of $K_\bullet$.

Emergent persistent pairs

A persistence pair (σ, τ) for a simplexwise filtration is

- an *emergent facet pair* if σ is the youngest facet of τ,
- an *emergent cofacet pair* if τ is the oldest cofacet of σ.

Clearly, every apparent pair is both.

Proposition

Let $K_\bullet$ be the lexicographically refined Rips filtration. Assume that

- *τ is the lexicographically minimal cofacet of σ with $\operatorname{diam}(\tau) = \operatorname{diam}(\sigma)$, and*
- *τ is not already in a persistence pair (ρ, τ) of $K_\bullet$ with $\rho \succ \sigma$.*

Then (σ, τ) is a 0-persistence emergent coface pair of $K_\bullet$.

- Includes all apparent pairs with persistence 0 (most pairs are of this kind)
- Can be identified *without* enumerating all cofaces of σ (shortcut for computation)

Speedup from emergent pairs shortcut

Previous example ($k = 2, n = 192$):

- Only $53 + 601 = 654$ out of $18\,337 + 1\,179\,808$ columns fully constructed without shortcut
- Only $164\,214 + 3\,392\,039 = 3\,556\,253$ out of $3\,447\,550 + 216\,052\,515 = 219\,500\,065$ nonzero entries of the coboundary matrix are enumerated

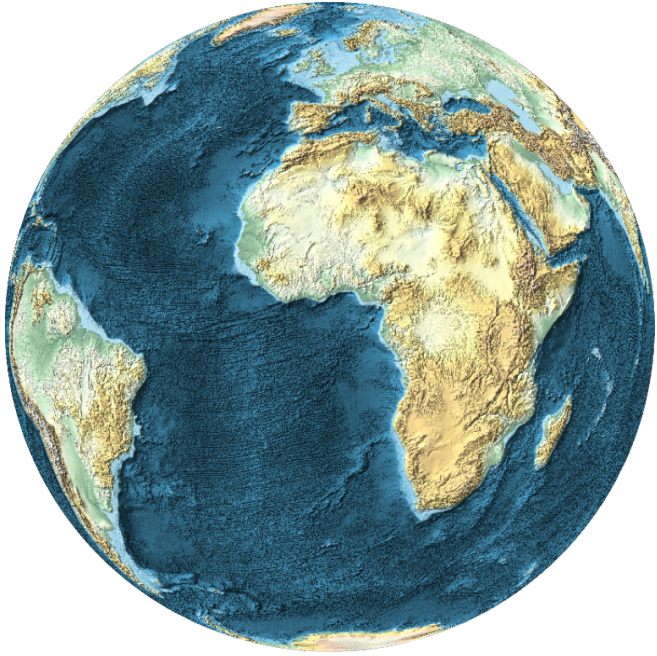
Speedup: $1.2\,s$ vs. $5.6\,s$ (factor 4.7)

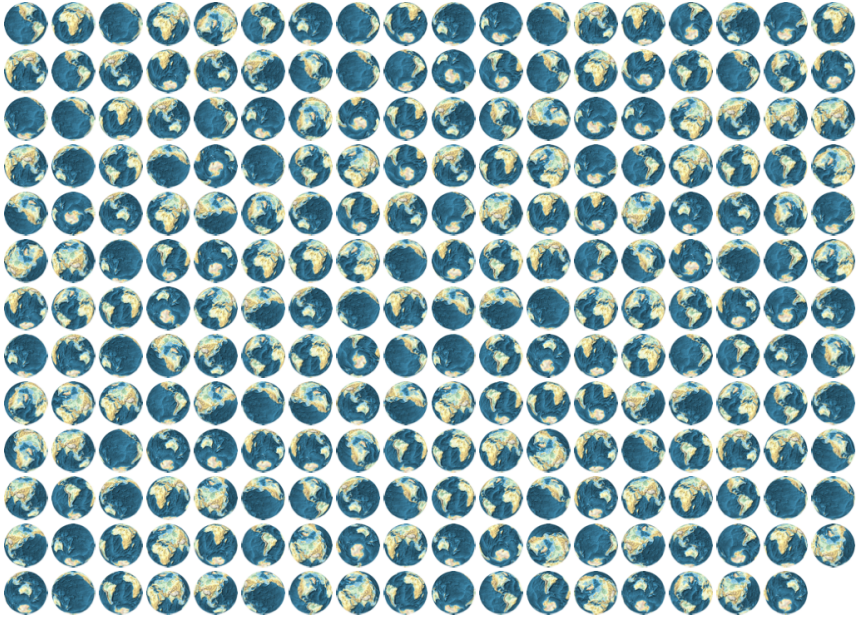
Apparent pairs in dimension 1

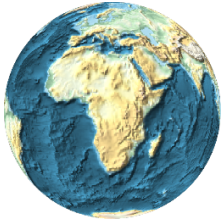
Theorem

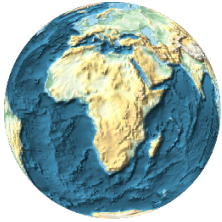
Let (X, d) be a finite metric space with distinct pairwise distances,
and let $K_\bullet$ be a simplexwise refinement of the Vietoris–Rips filtration for X.
In dimension 1, the zero persistence pairs of $K_\bullet$ are precisely the apparent pairs.

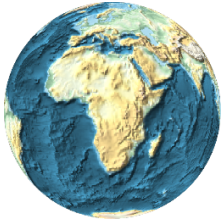
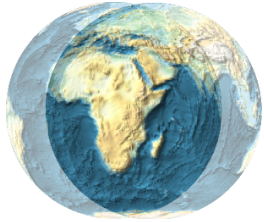
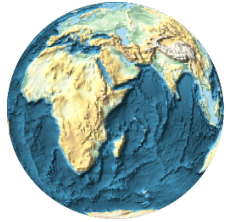
Around the world

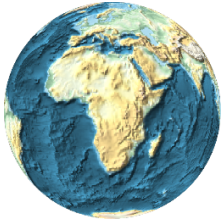
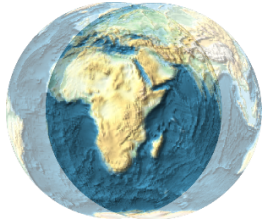
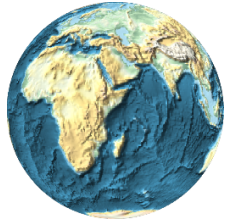




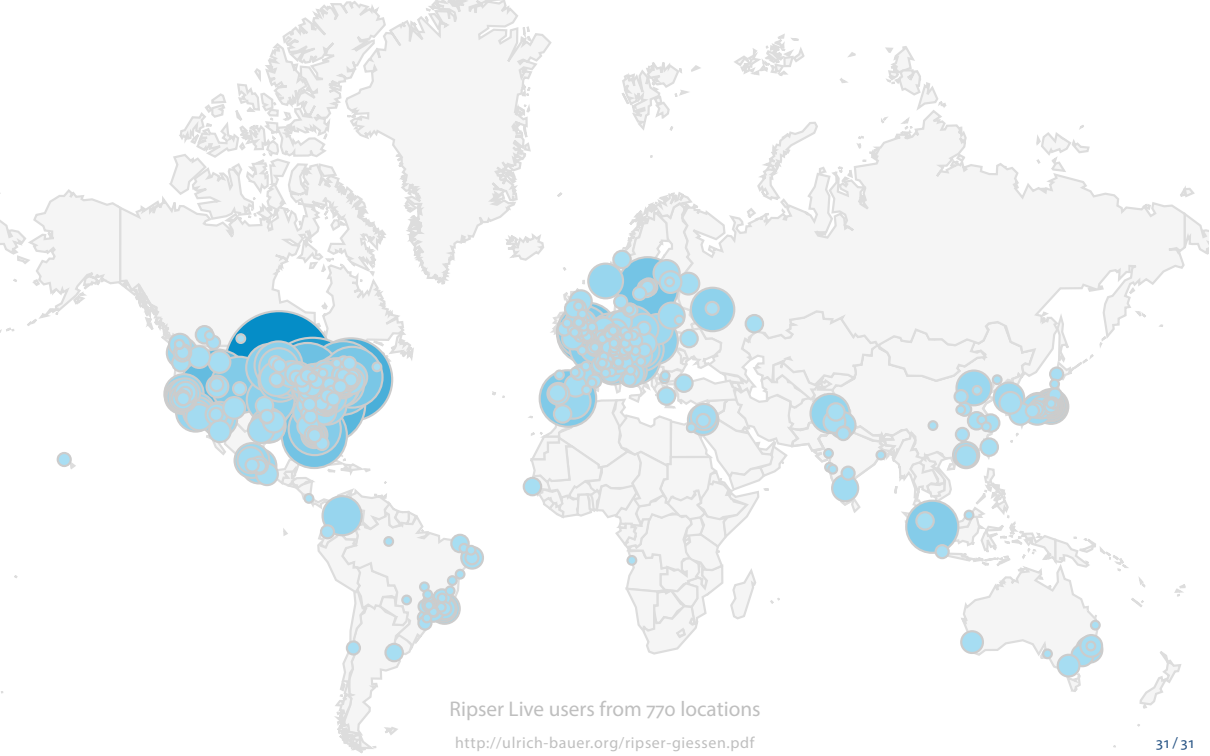




Demo (Ripser)



Ripser Live users from 770 locations