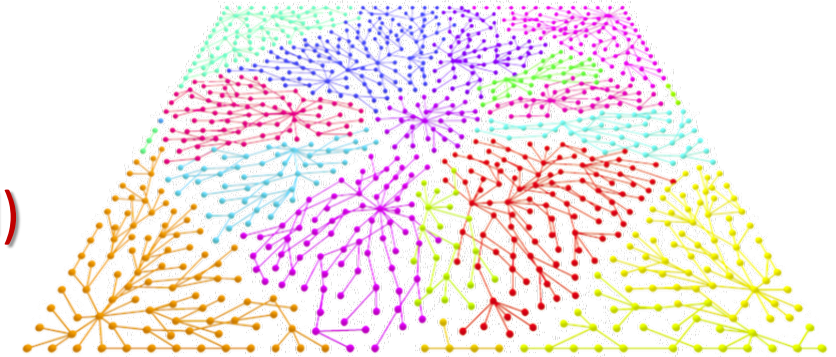
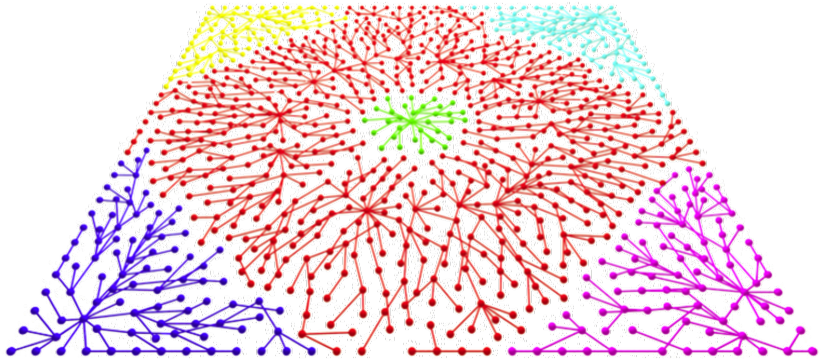
CS233, CME251: Geometric and Topological Data Analysis

Leonidas Guibas
Computer Science Department
Stanford University

Guest lecture: Samir Chowdhury (Stanford)

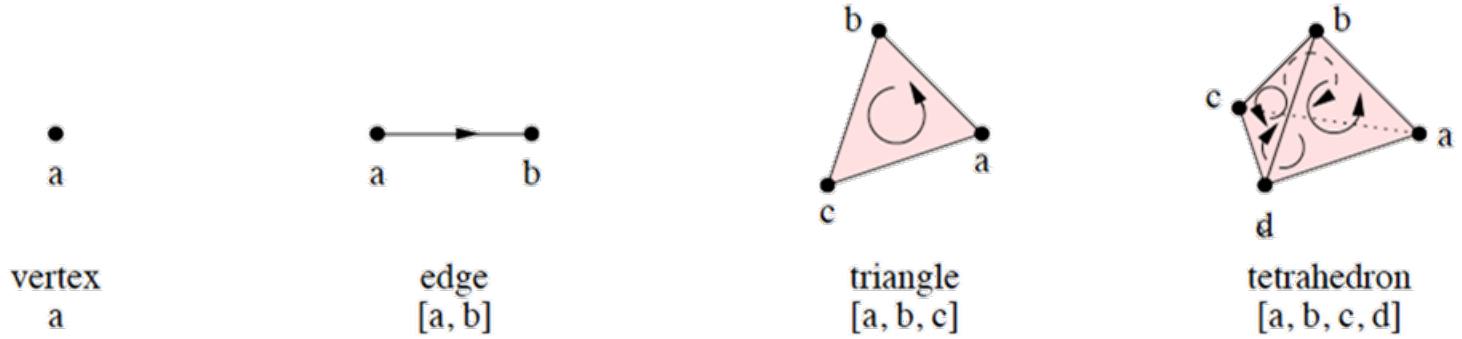




Lecture 8
29 April 2020

Last Time: From topology to (linear) algebra

Recall: simplicial complexes



- An (abstract) simplicial complex Σ on a vertex set K is a collection of nonempty, finite subsets of K (called simplices) that is closed under inclusion. In other words, whenever $\emptyset \neq \tau \subseteq \sigma \in \Sigma$, we also have $\tau \in \Sigma$. In this case, τ is called a face of σ.
- Notation: we'll reduce notation and write simplices as *a*, *ab*, *abc*, and *abcd*.

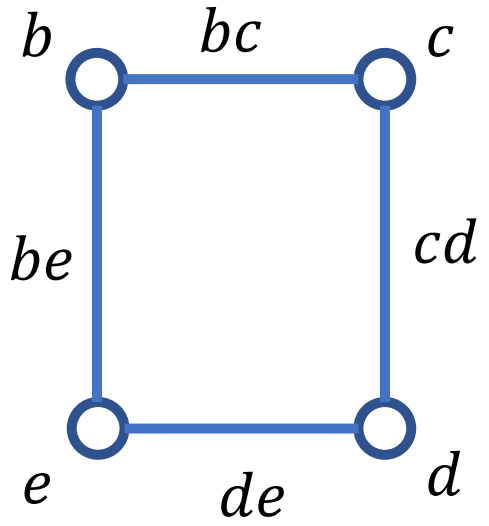
Recall: simplicial complexes → vector spaces

- $\mathbb{F}$ a field, typically $\mathbb{Z}/2\mathbb{Z} = \{0,1\}$
- Σ a simplicial complex, Σ_0 the 0-simplices (vertices), Σ_1 the 1-simplices (edges), Σ_2 the 2-simplices (triangles), …
- Algebraic gadget: the free vector spaces $\mathbb{F}[\Sigma_0], \mathbb{F}[\Sigma_1], \mathbb{F}[\Sigma_2], \ldots$
  - Intuitively, just allow addition and scalar multiplication of simplices!
  - Linear combinations of p-simplices will be called p-chains
- Standard notation: $C_p(\Sigma)$, the vector space of p-chains

Recall: simplicial complexes → vector spaces

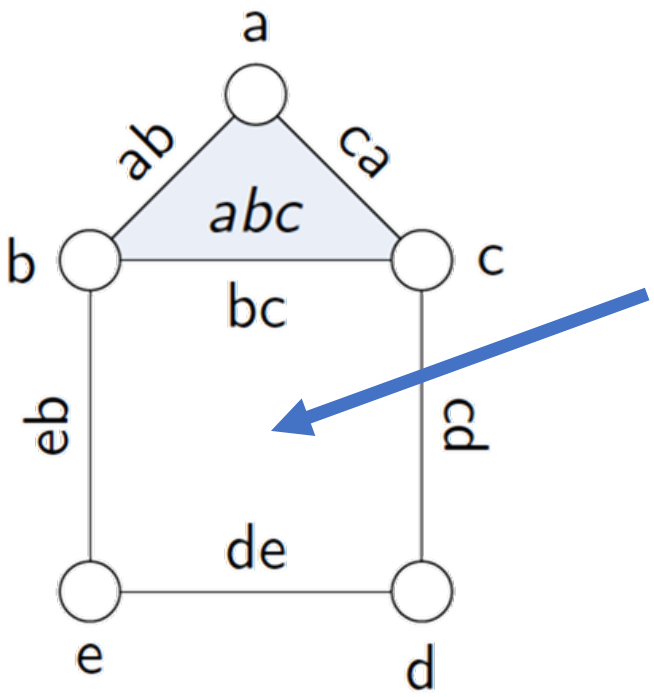
- Σ a simplicial complex, Σ_0 the 0-simplices (vertices), Σ_1 the 1-simplices (edges), Σ_2 the 2-simplices (triangles), …

- $C_p(\Sigma)$, the vector space of p-chains



- Intuitively, $bc + cd + de + be$ traces out a loop bounding a box. Can this be formalized?

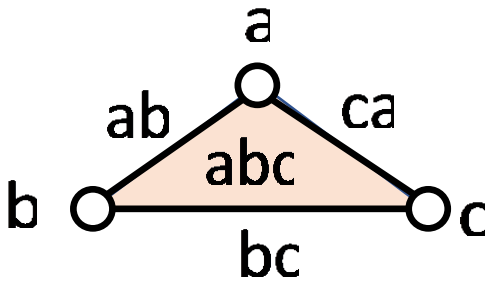
Recall: the boundary map ∂



- Alternating sum (hat denotes omission):

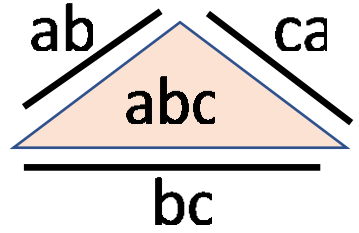
$$\partial[x_0, \dots x_n] = \sum_{i=0}^{n} (-1)^i \, [x_0, \dots, \hat{x}_i, \dots x_n]$$

$$\partial(bc + cd + de + eb)$$
$$= c - b + d - c + e - d + b - e = 0.$$



- Key identity: $\partial^2 = 0$.

$$\partial^2(abc) = \partial(bc - ac + ab)$$
$$= c - b - c + a + b - a = 0.$$



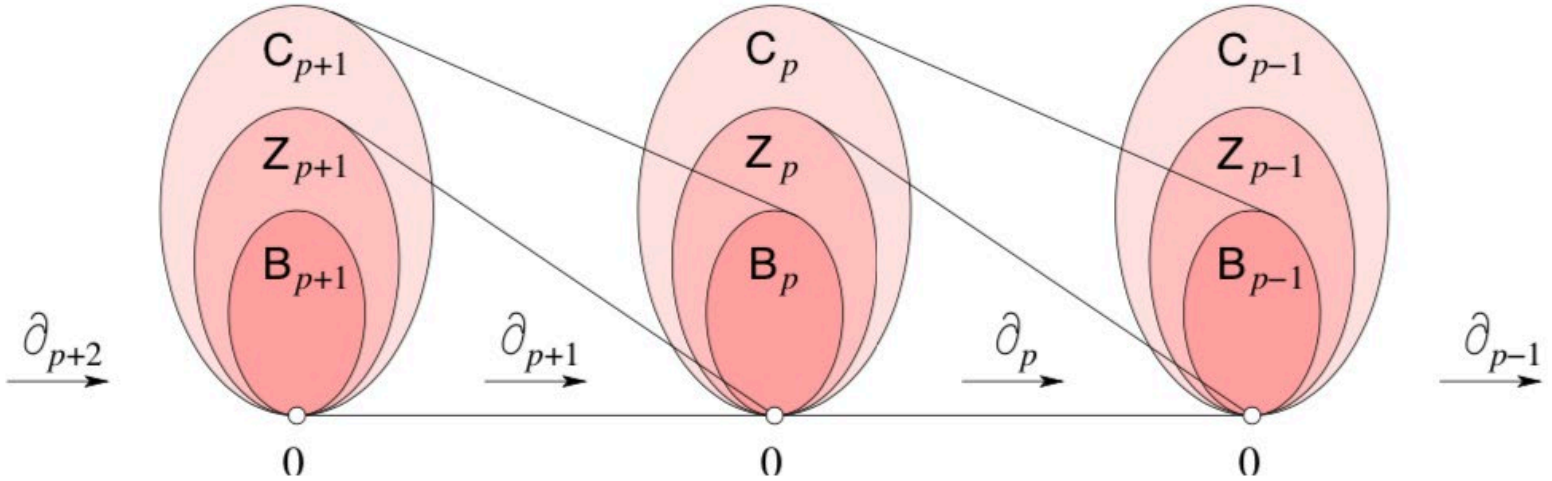
- $\partial = 0$ on linear combinations that are boundaries. Intuitively, $\partial^2 = 0$ says that the boundary of a simplex really is a boundary!

Recall: the chain complex

- We now have a sequence of vector spaces and linear maps:

$$\dots \xrightarrow{\partial_{p+2}} C_{p+1} \xrightarrow{\partial_{p+1}} C_p \xrightarrow{\partial_p} C_{p-1} \xrightarrow{\partial_{p-1}} \dots$$

- Notation: $Z_p \coloneqq \ker \partial_p = \{v \in C_p : \partial_p(v) = 0\}$ the p-cycles, $B_p \coloneqq im\ \partial_{p+1} \subseteq C_p$ the p-boundaries.



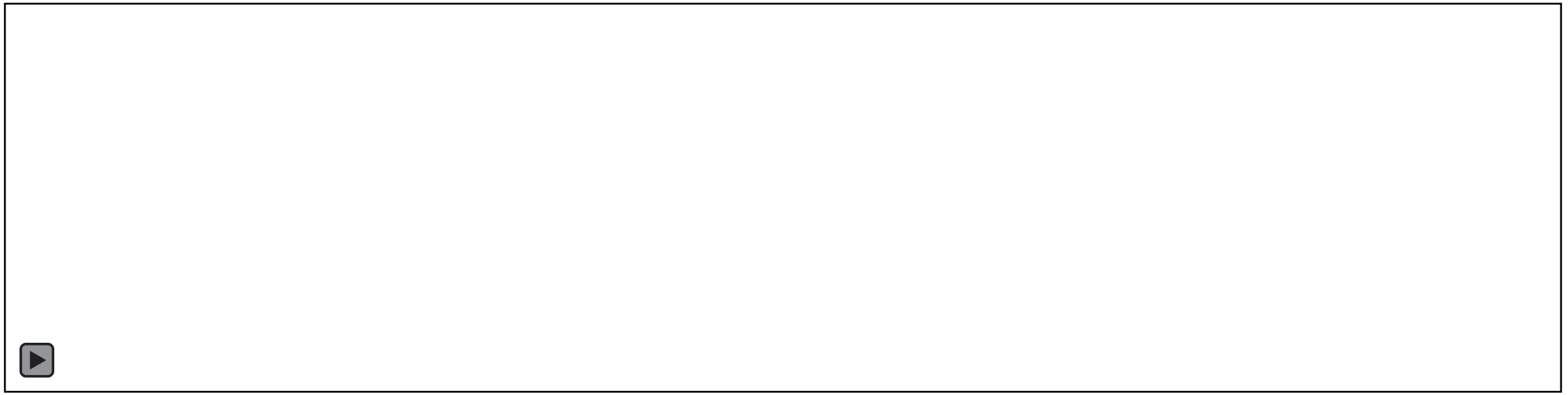
pth homology vector space:

$$H_p \coloneqq \frac{\ker \partial_p}{im\ \partial_{p+1}}$$

Intuition: p-dimensional connectivity

Today: persistent homology

Youtube “**Persistent Homology on a noisy torus**”



$$\emptyset \subseteq \Sigma^{(1)} \subseteq \Sigma^{(2)} \subseteq \ \ldots \subseteq \Sigma^{(n-1)} \subseteq \Sigma^{(n)}$$

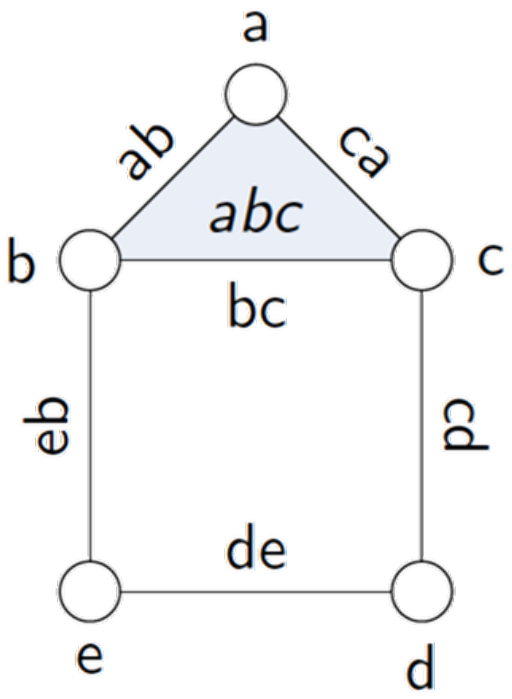
Functoriality of homology

$$0 \ \rightarrow H_p\left(\Sigma^{(1)}\right) \ \rightarrow \ H_p(\Sigma^{(2)}) \ \rightarrow \ \ldots \rightarrow \ H_p(\Sigma^{(n-1)}) \ \rightarrow \ H_p(\Sigma^{(n)})$$

Background: Computing homology

Setting up the linear algebra

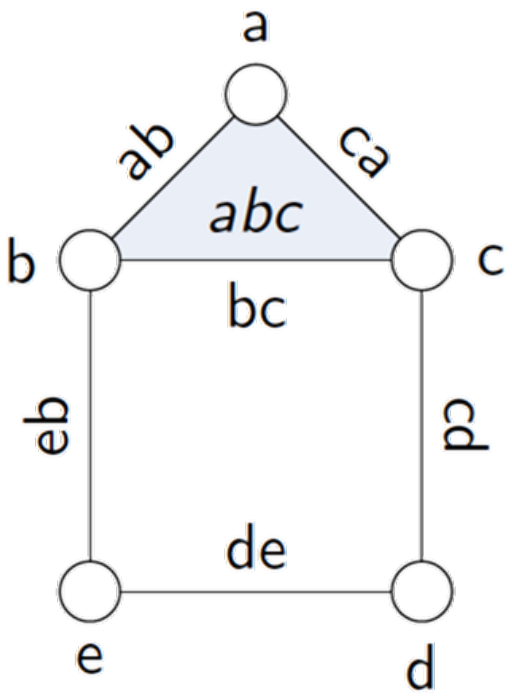
- $\partial_p : C_p \to C_{p-1}$ is a linear map between vector spaces, so we'll use matrices



$$\partial_1 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccccc} ab & ca & bc & cd & de & eb \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

$$\partial_2 = \begin{array}{c} \\ ab \\ ca \\ bc \\ cd \\ de \\ eb \end{array} \begin{array}{c} \begin{array}{c} abc \end{array} \\ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array}$$

Setting up the linear algebra

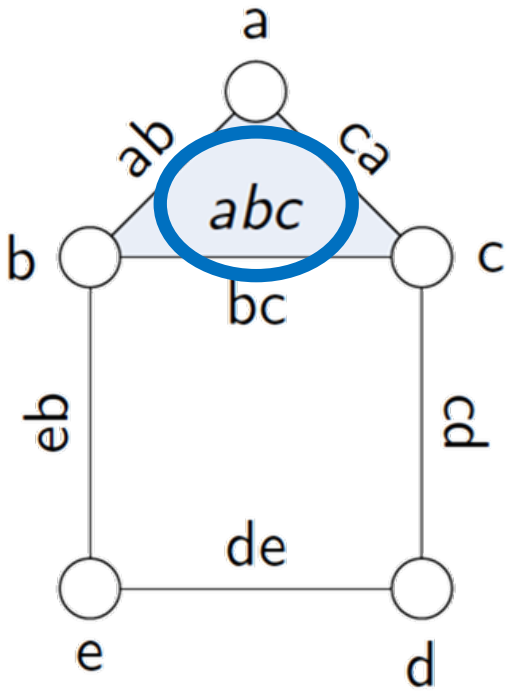


$$\partial_1 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccccc} ab & ca & bc & cd & de & eb \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

$$\partial_2 = \begin{array}{c} \\ ab \\ ca \\ bc \\ cd \\ de \\ eb \end{array} \begin{array}{c} \begin{array}{c} abc \end{array} \\ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array}$$

- E. g. $H_1 = \frac{\ker \partial_1}{im\ \partial_2}$, so we need to extract bases for these vector spaces

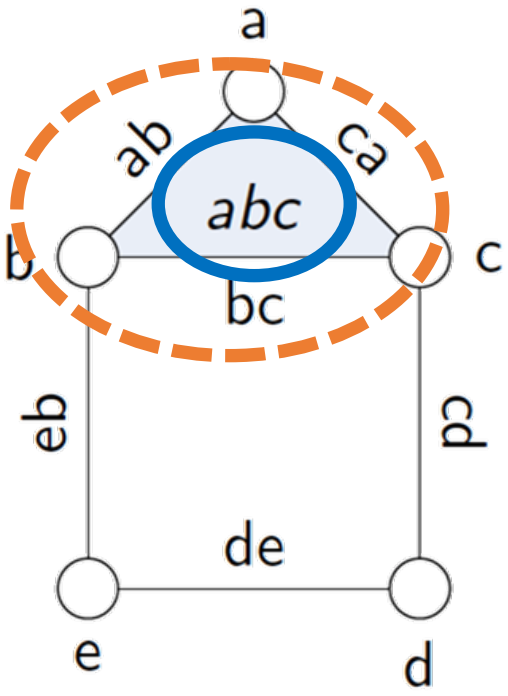
Setting up the linear algebra



$$\partial_1 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccccc} ab & ca & bc & cd & de & eb \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

$$\partial_2 = \begin{array}{c} abc \\ \begin{array}{c} ab \\ ca \\ bc \\ cd \\ de \\ eb \end{array} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array}$$

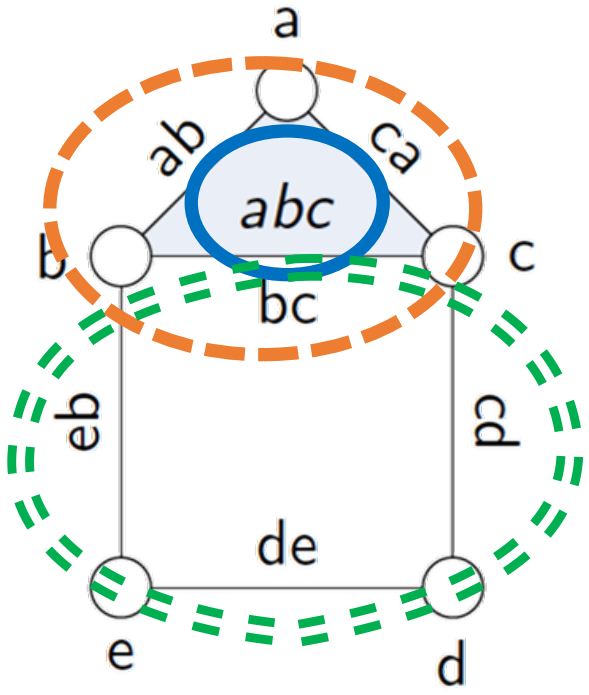
Setting up the linear algebra



$$\partial_1 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccccc} ab & ca & bc & cd & de & eb \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

$$\partial_2 = \begin{array}{c} \\ ab \\ ca \\ bc \\ cd \\ de \\ eb \end{array} \begin{array}{c} \begin{array}{c} abc \end{array} \\ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array}$$

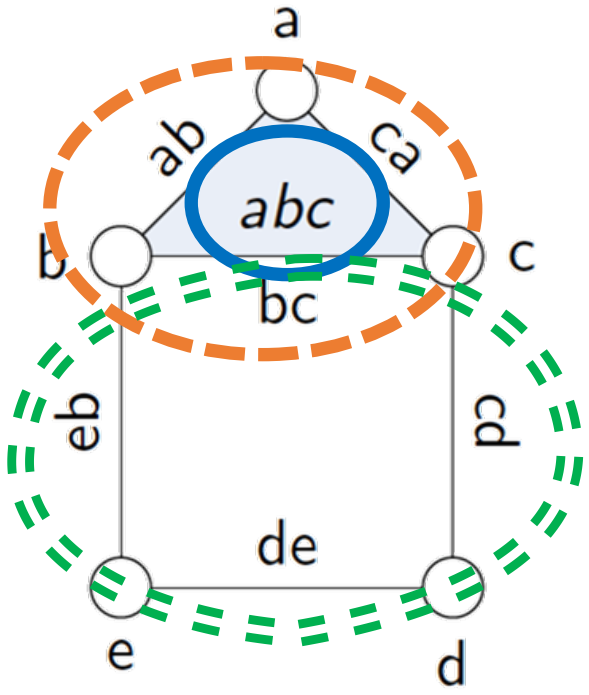
Setting up the linear algebra



$$\partial_1 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccccc} ab & ca & bc & cd & de & eb \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

$$\partial_2 = \begin{array}{c} \begin{array}{c} abc \end{array} \\ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array} \begin{array}{l} ab \\ ca \\ bc \\ cd \\ de \\ eb \end{array}$$

Setting up the linear algebra

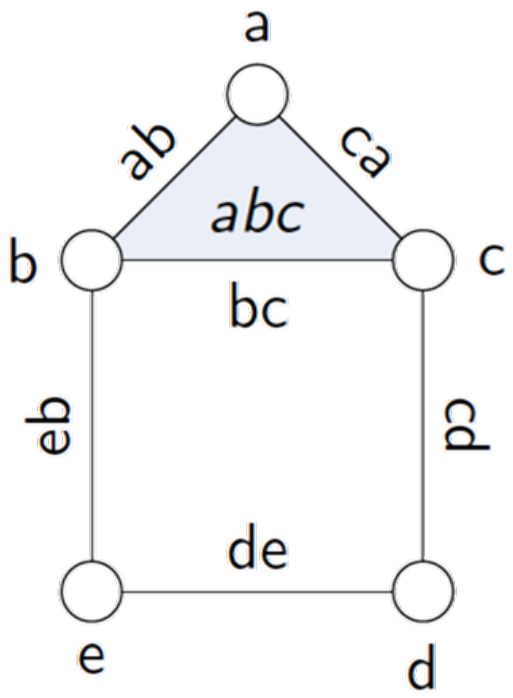


$$\partial_1 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccccc} ab & ca & bc & cd & de & eb \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

$$\partial_2 = \begin{array}{c} \begin{array}{c} abc \end{array} \\ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array} \begin{array}{l} ab \\ ca \\ bc \\ cd \\ de \\ eb \end{array}$$

- Need $\mathbb{Z}/2\mathbb{Z}$ arithmetic
- Column operations

Summary so far



$$\ldots \to C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \to 0$$

{abc} $\xrightarrow{\partial_2}$ {ab,ca,bc,cd,de,eb} $\xrightarrow{\partial_1}$ {a,b,c,d,e}

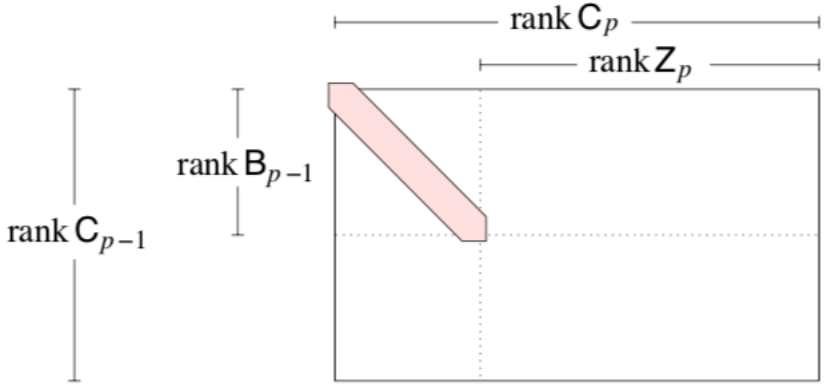
$$\partial_2 = \begin{array}{c} \\ ab \\ ca \\ bc \\ cd \\ eb \\ de \end{array} \begin{array}{c} \begin{array}{c} abc \end{array} \\ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{array}$$

$$\partial_1 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccccc} ab & ca & bc & cd & eb & de \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

Smith Normal Form

Standard algorithm for homology:

- Use row/column operations (Gaussian elimination) on boundary matrices to get form where initial segment of diagonal is 1, everything else 0



- Read off basis for $\ker \partial_p$ (nullspace) and $im\ \partial_{p+1}$(range)

- Diagonal form means we have 1-1 correspondence, know which cycles are boundaries

- We'll see a slightly different method that only uses column operations and goes through unchanged for *persistent* homology

Column operations in $\mathbb{Z}/2\mathbb{Z}$

Arithmetic in $\mathbb{Z}/2\mathbb{Z}$ is very easy:

$$\begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix}$$

$$\begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix}$$

$$\begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$$

Reduction algorithm for homology*

*Cohen-Steiner, Edelsbrunner, Morozov - **Vines and Vineyards by Updating Persistence in Linear Time**

To get H_p, we need: (1) a basis for $\ker \partial_p$, and (2) a sub-basis for $im\ \partial_{p+1}$.

	ab	ca	bc	cd	de	eb	abc
a	1	1	0	0	0	0	0
b	1	0	1	0	0	1	0
c	0	1	1	1	0	0	0
d	0	0	0	1	1	0	0
e	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0

R

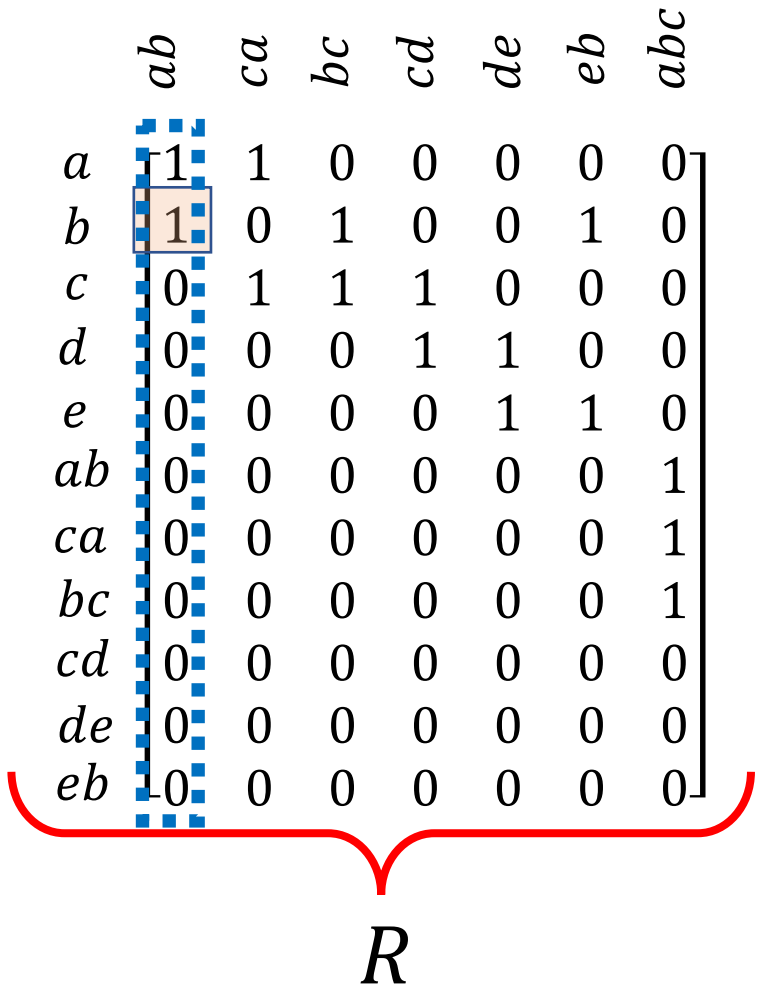
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R \coloneqq D$

For column $j = 1, 2, \dots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



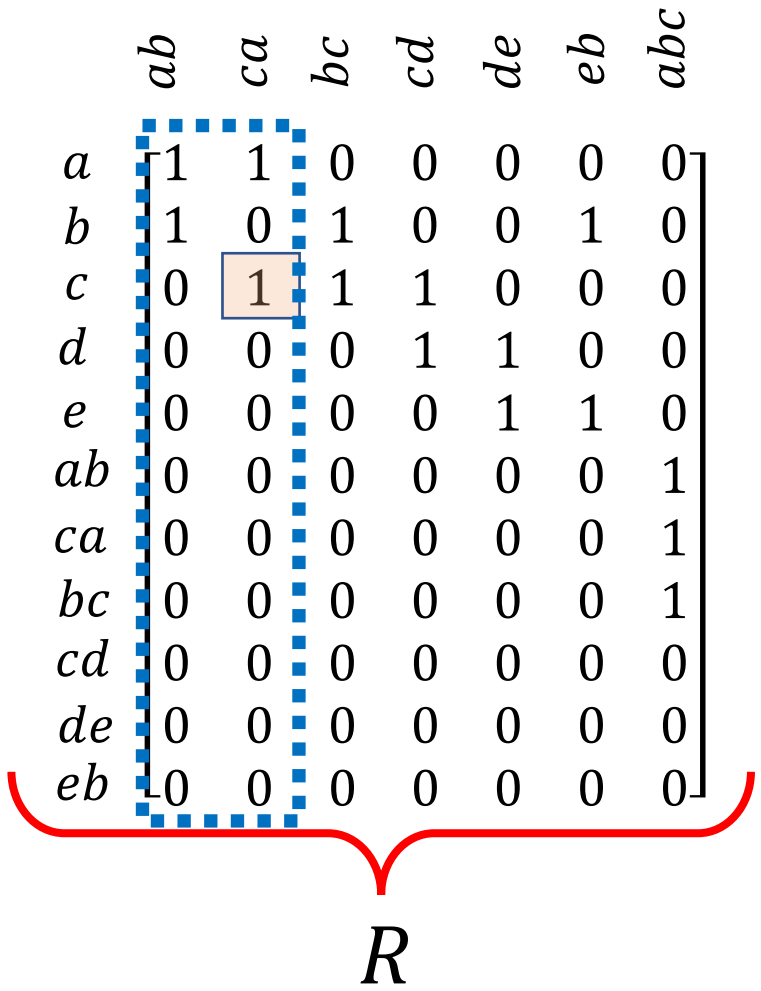
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R := D$

For column $j = 1, 2, \ldots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



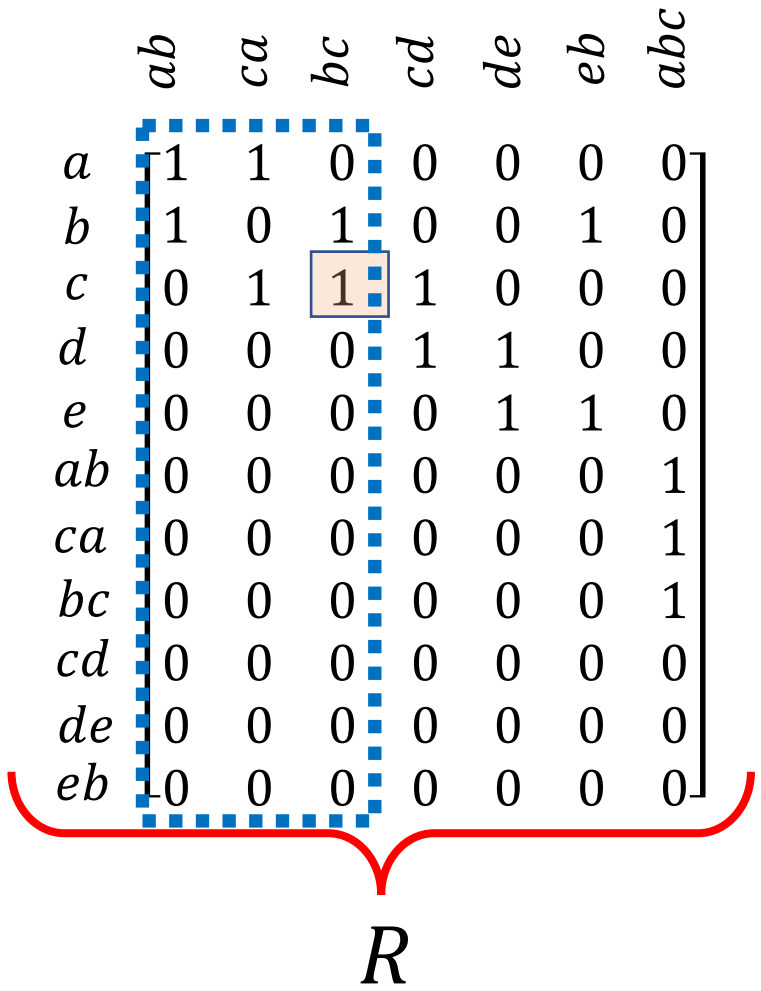
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R := D$

For column $j = 1, 2, \ldots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



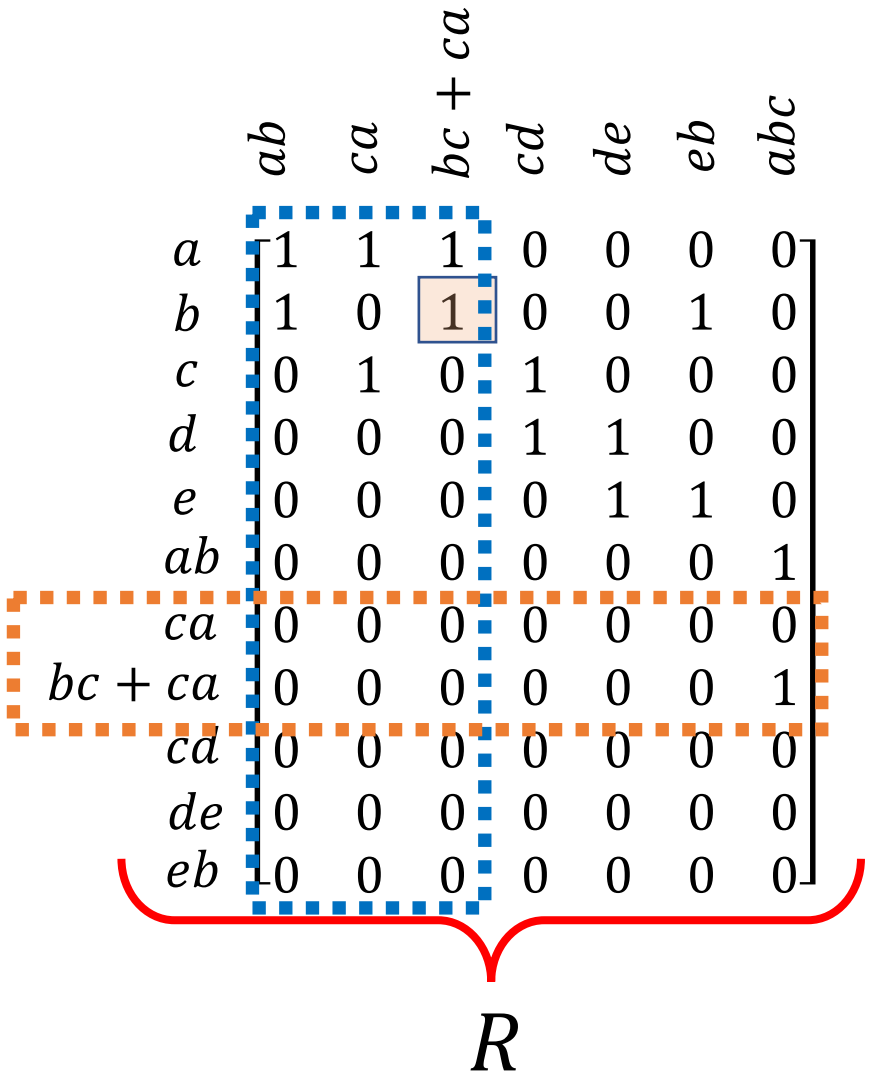
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R := D$

For column $j = 1, 2, \ldots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



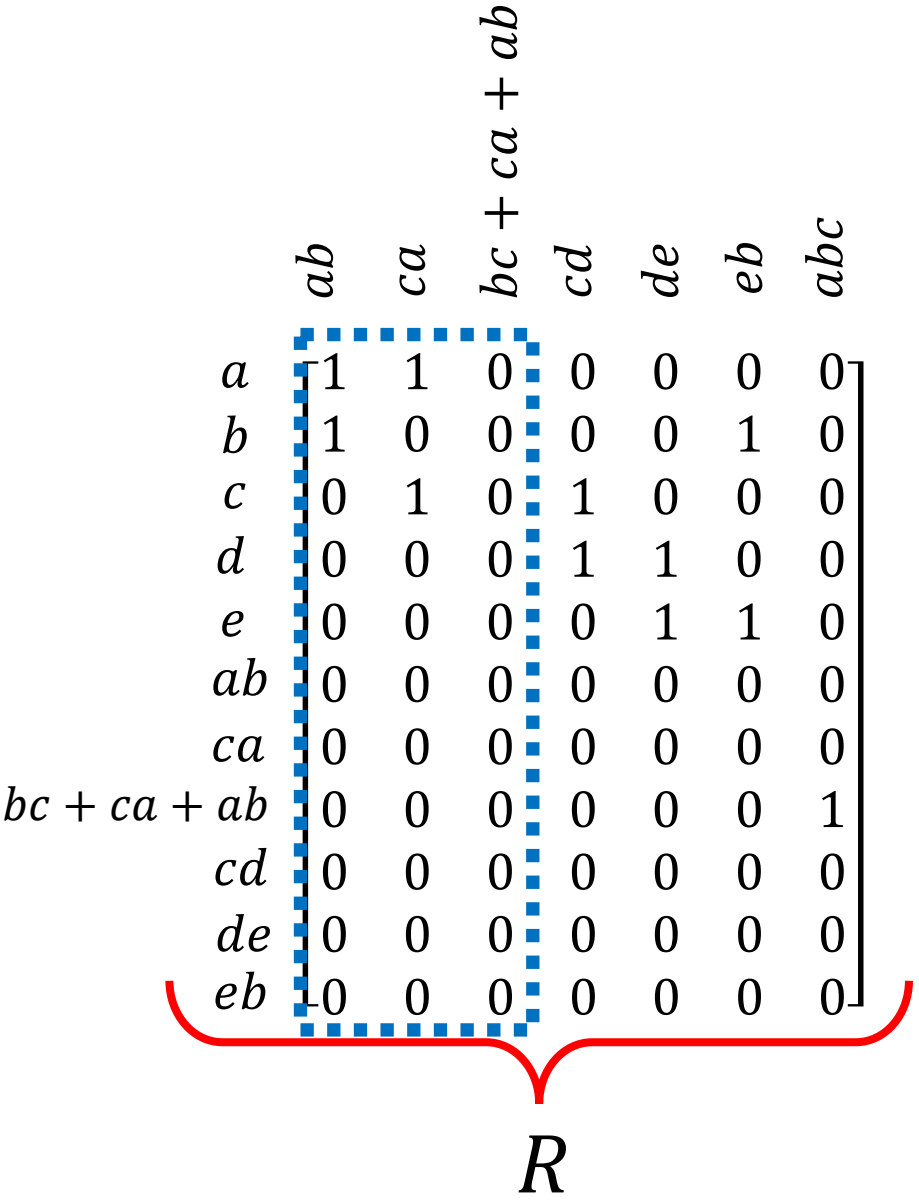
Reminder: to keep bases compatible, column operations require corresponding row operation!

E.g. $T: V \to W$ linear map, $\{v_1, v_2\}, \{w_1, w_2\}$ bases, $T(v_1) = a_{11}w_1 + a_{21}w_2$. Changing codomain basis by setting $w_1 \leftarrow w_1, w_2 \leftarrow w_2 + w_1$ means we have:

$$T(v_1) = a_{11}w_1 + a_{21}(w_2 + w_1) - a_{21}w_1$$
$$= (a_{11} - a_{21})w_1 + a_{21}(w_2 + w_1).$$

I.e. subtract row 2 from row 1 (notice the "reflected" operation). In $\mathbb{Z}/2\mathbb{Z}$, this is just addition!

Reduction algorithm for homology



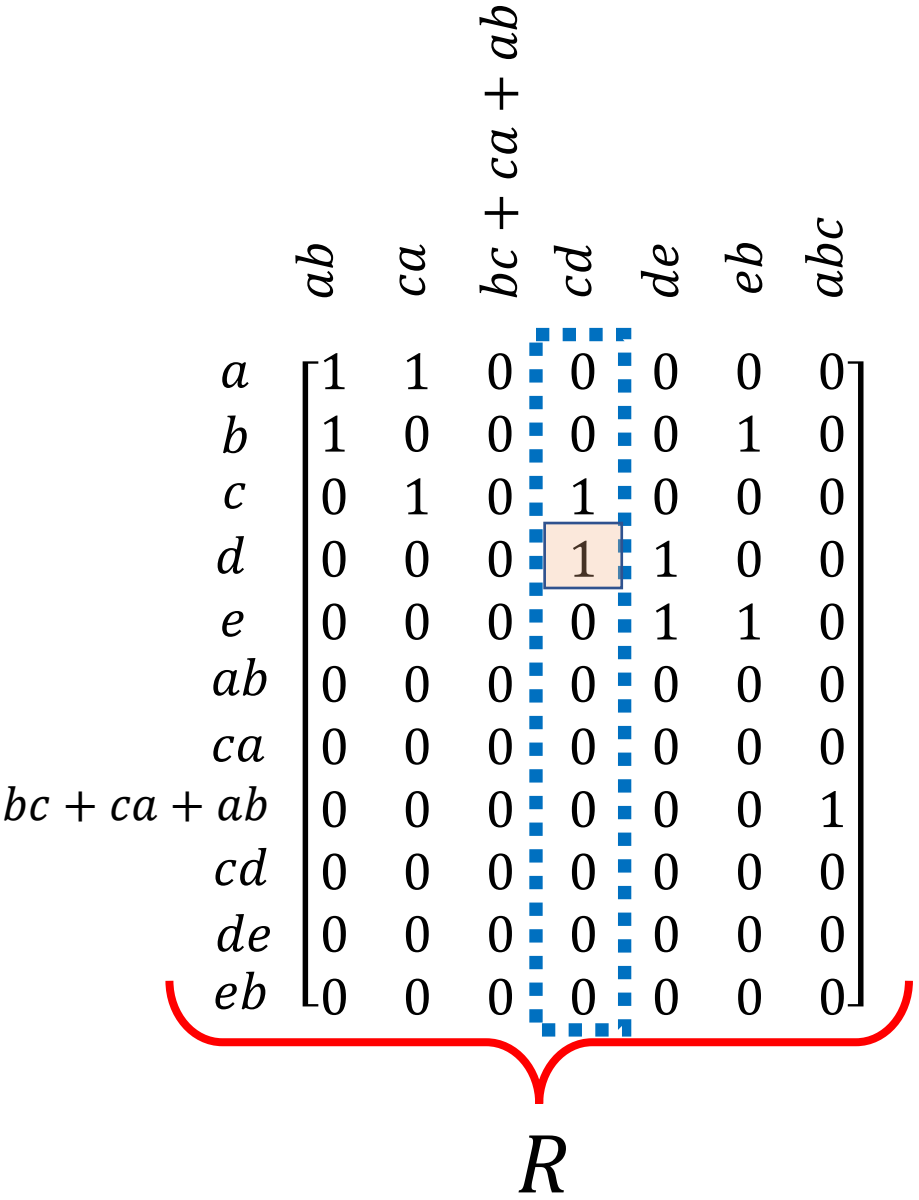
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R := D$

For column $j = 1, 2, \ldots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



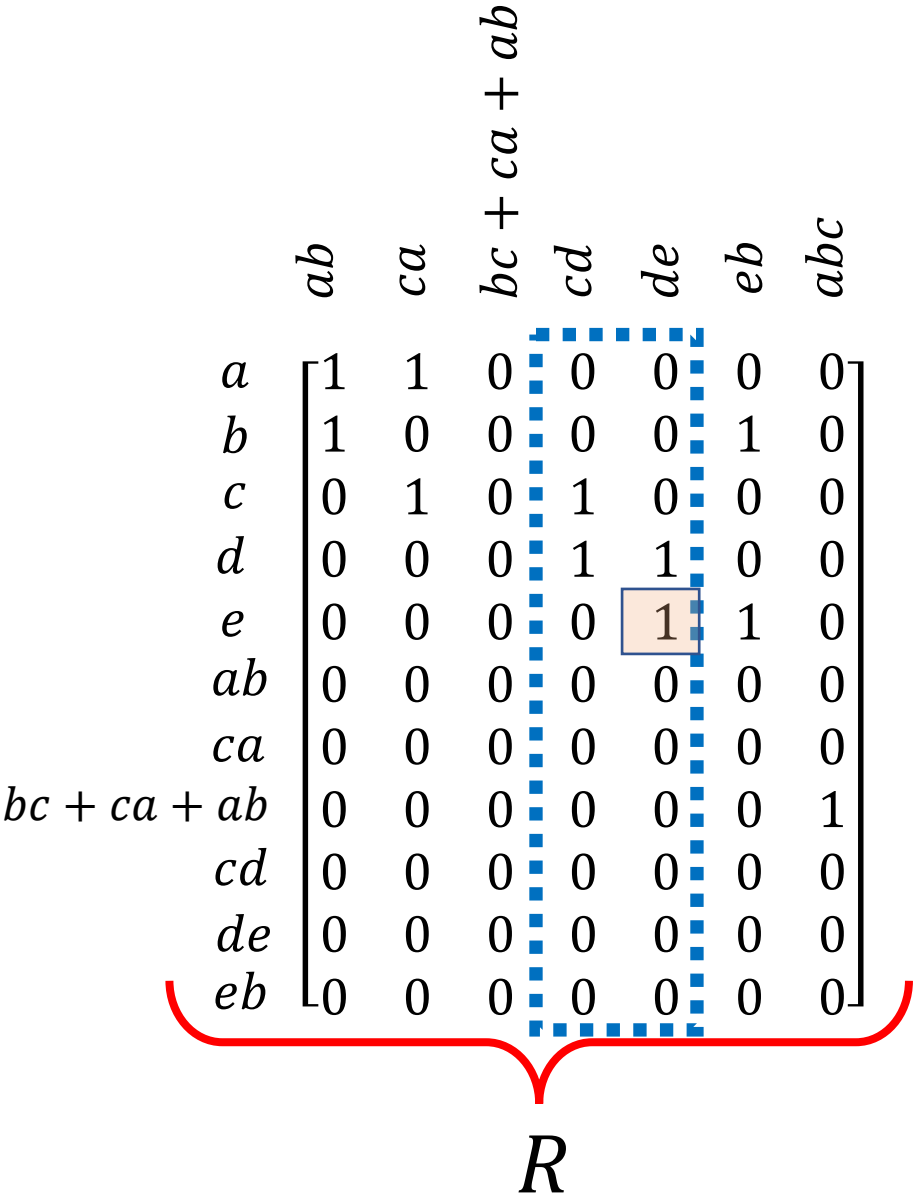
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R \coloneqq D$

For column $j = 1, 2, \ldots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



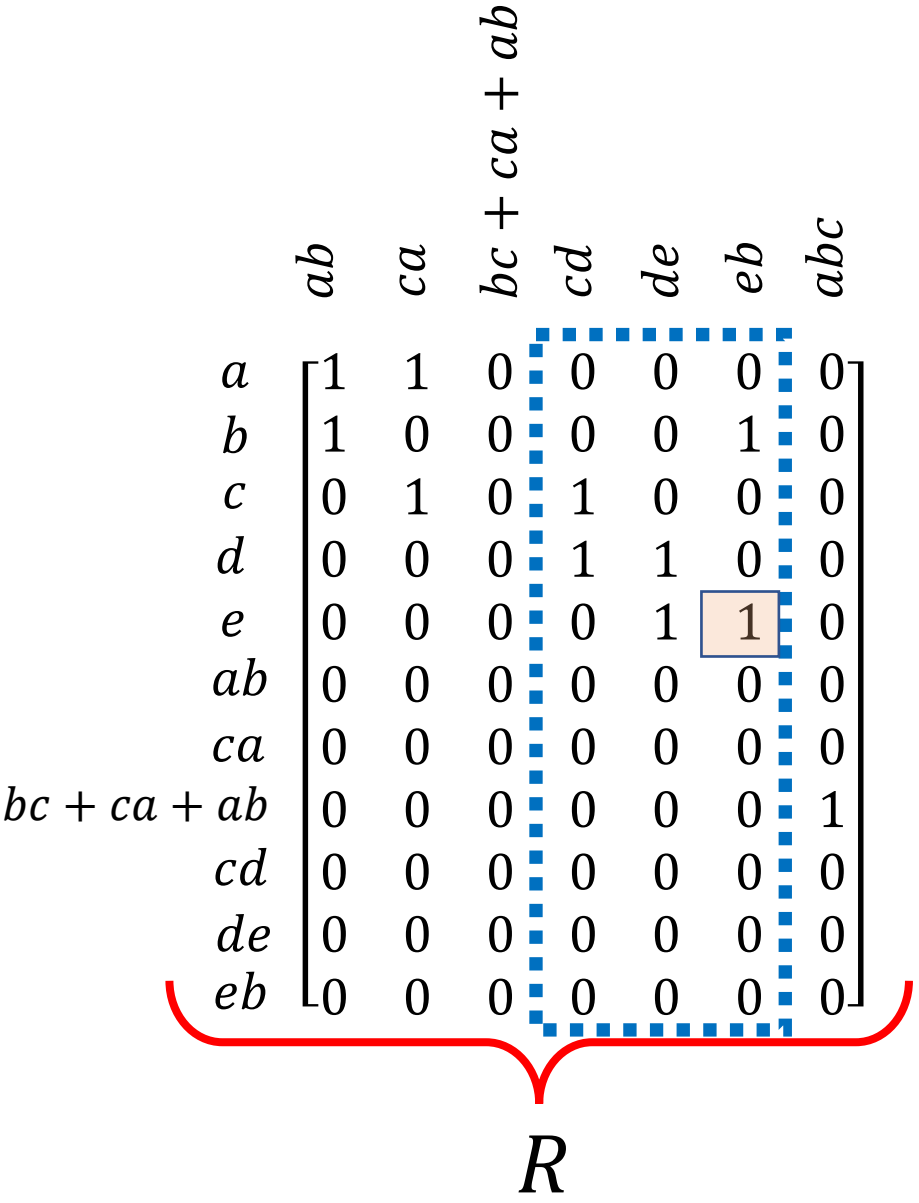
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R \coloneqq D$

For column $j = 1, 2, \ldots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



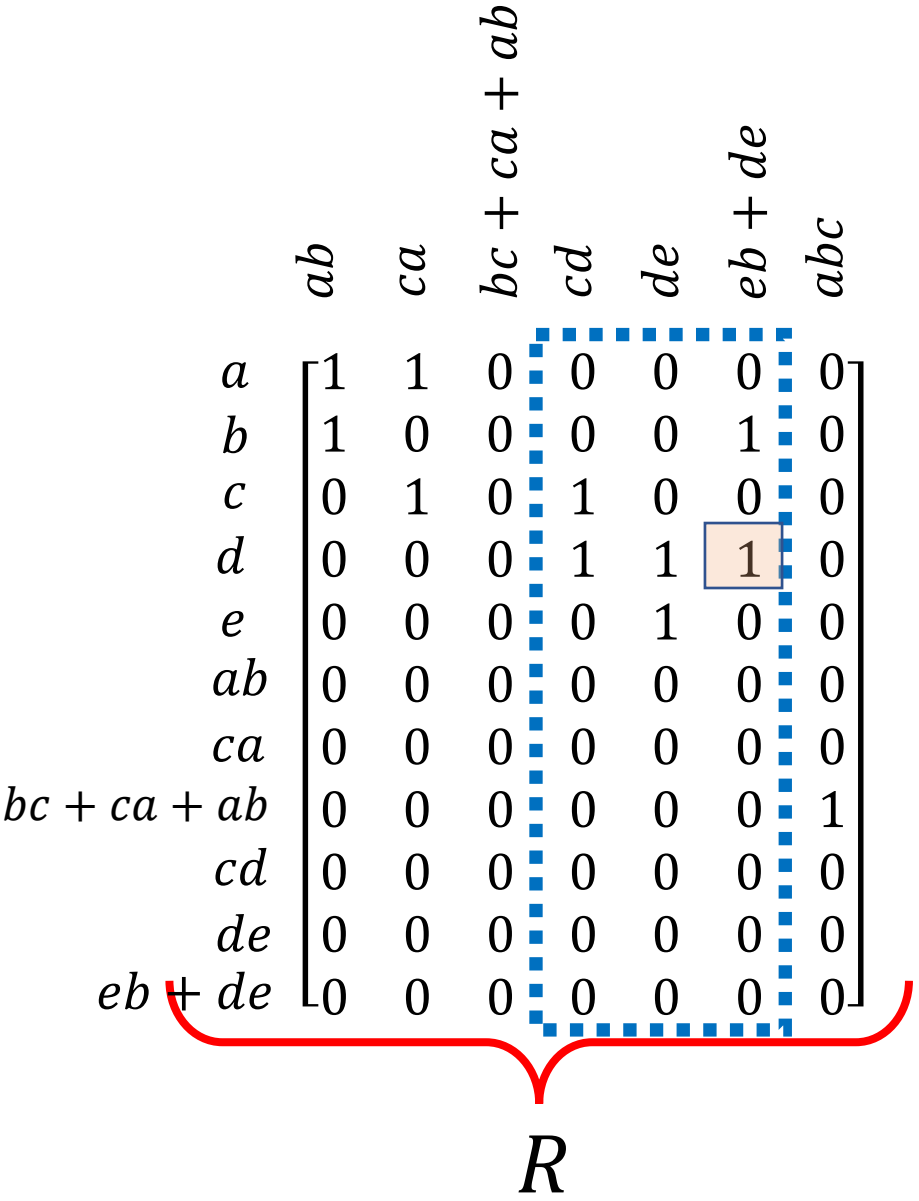
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R \coloneqq D$

For column $j = 1, 2, \ldots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



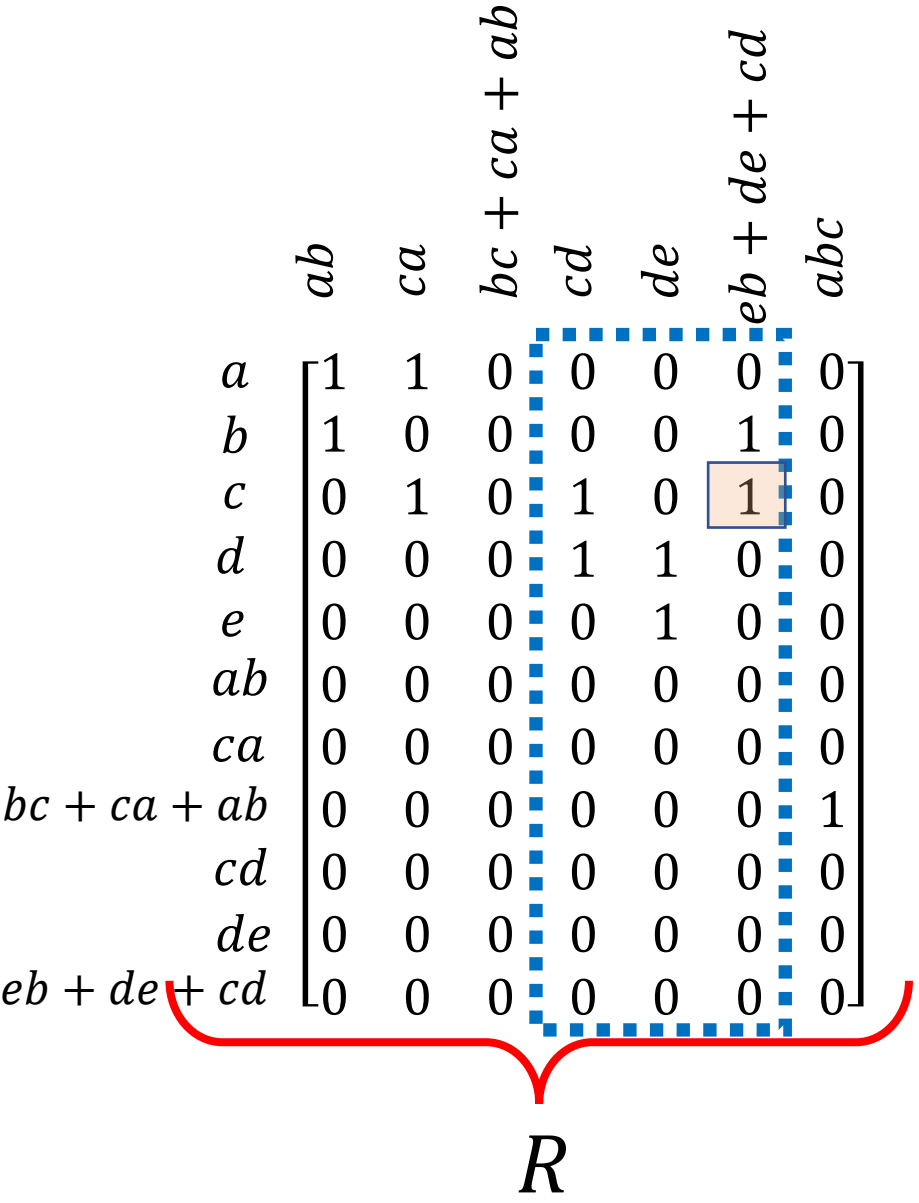
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R \coloneqq D$

For column $j = 1, 2, \ldots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



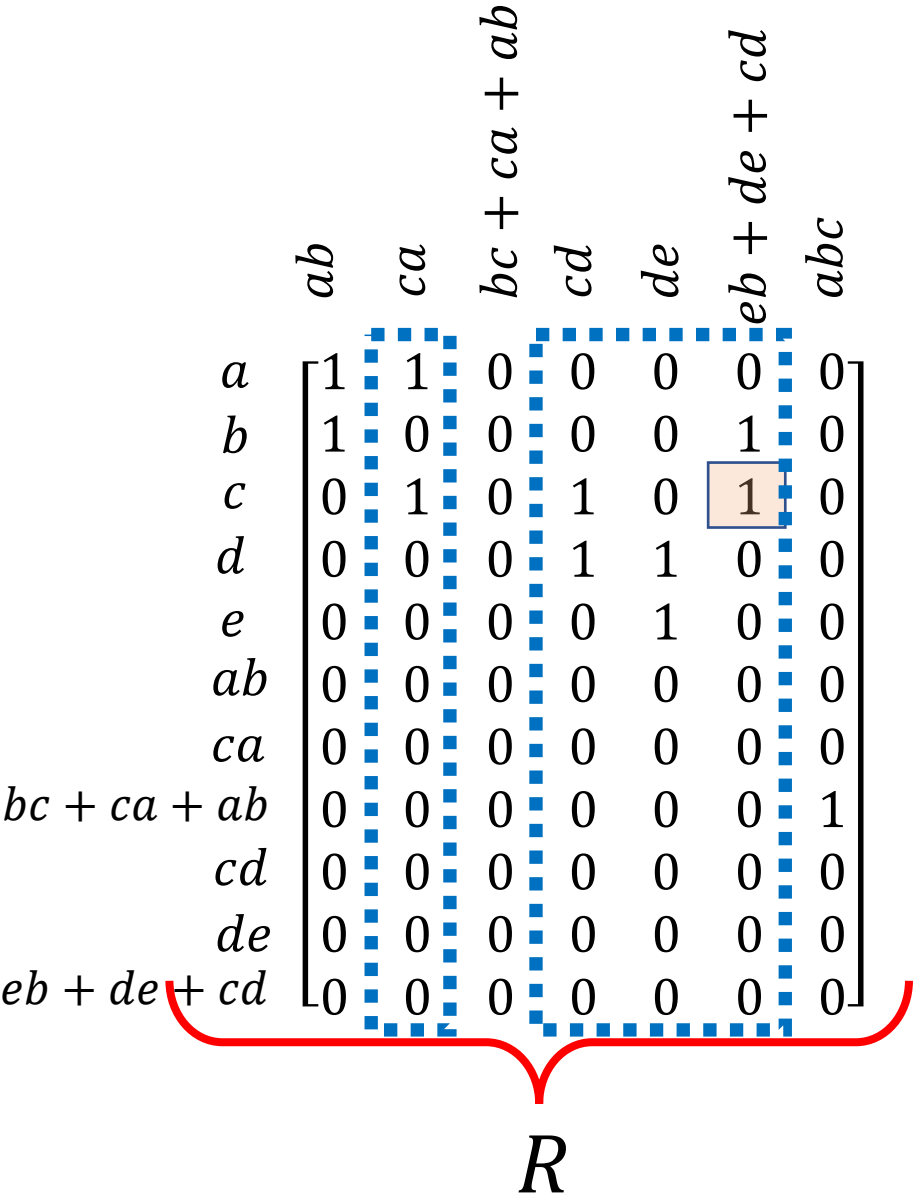
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R := D$

For column $j = 1, 2, \ldots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



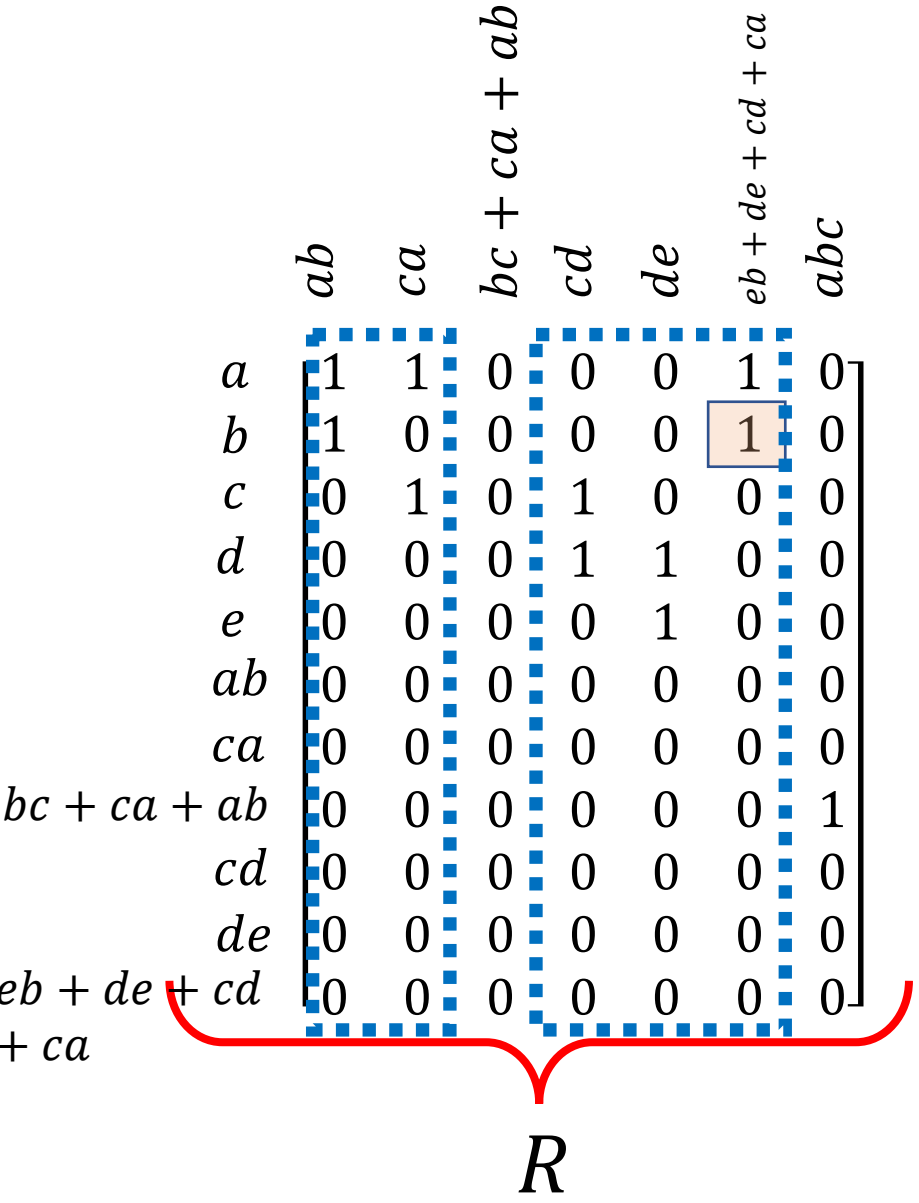
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R \coloneqq D$

For column $j = 1, 2, \ldots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



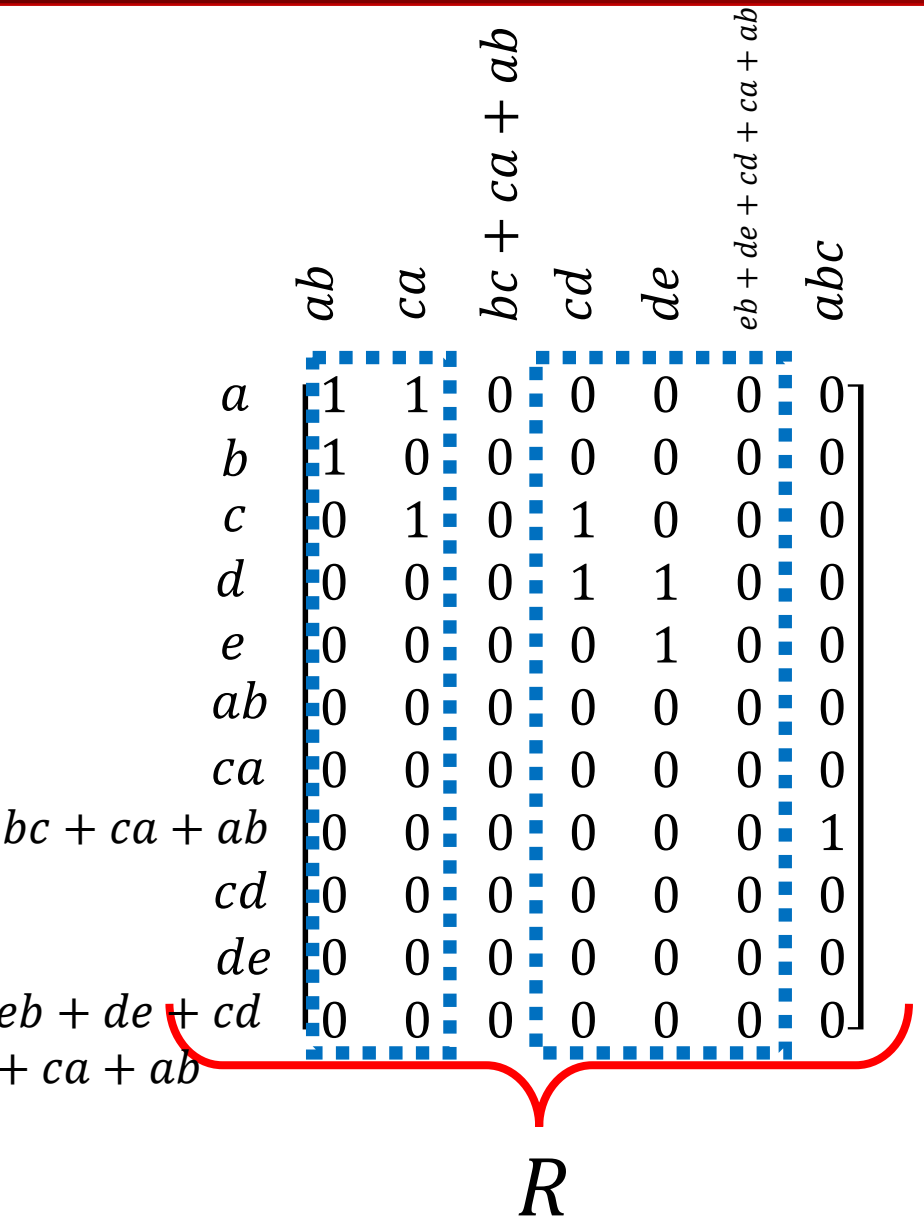
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R \coloneqq D$

For column $j = 1, 2, \ldots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



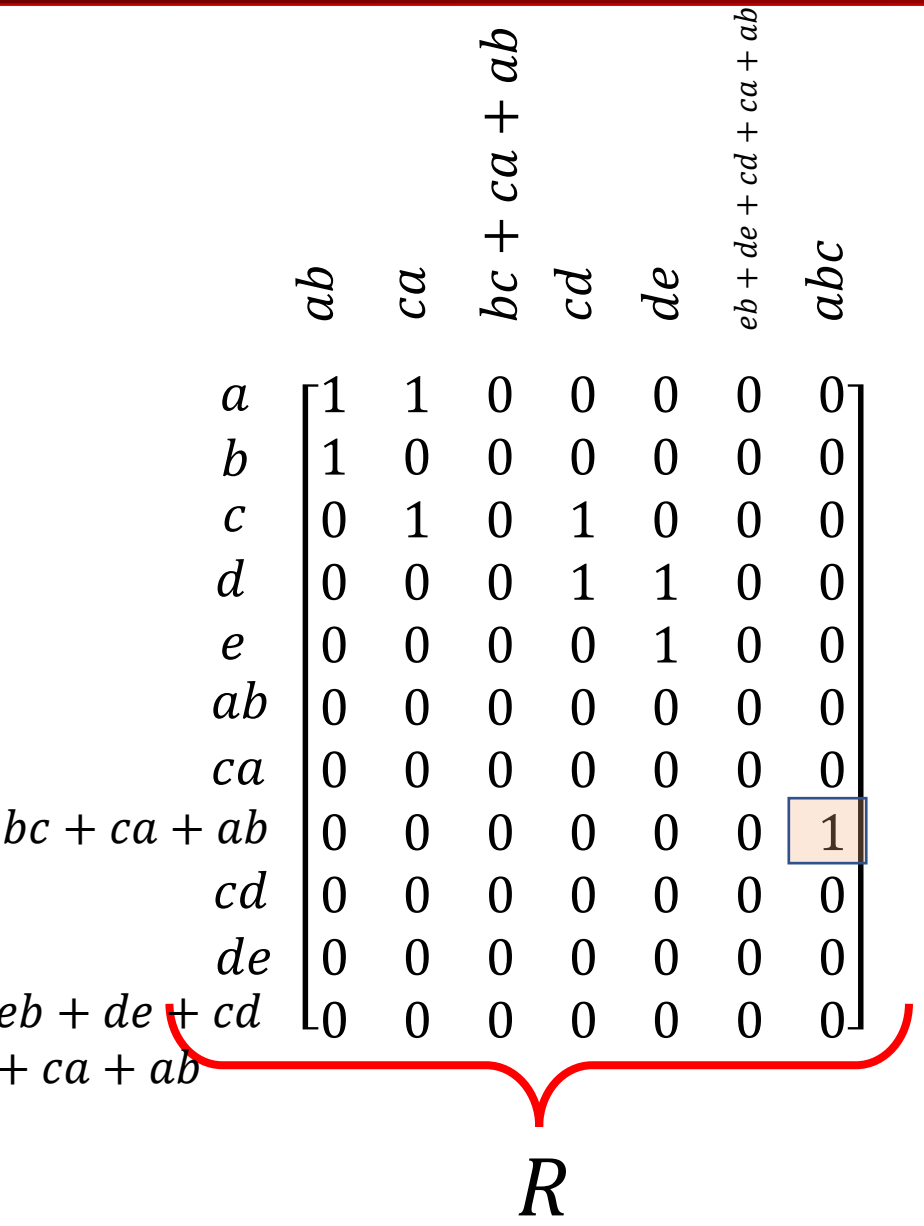
Notation: $low_R(j)$= row index of bottom 1 in column j of matrix R

Algorithm:

Initialize boundary information in incidence matrix D
Initialize $R := D$

For column $j = 1, 2, \ldots, n$
  while $\exists j' < j, low_R(j') = low_R(j)$ do
    add column j' to column j

Reduction algorithm for homology



Reduction algorithm for homology

	a	b	c	d	e	ab	ca	$bc+ca+ab$	cd	de	$eb+de+cd+ca+ab$	abc
a	0	0	0	0	0	1	1	0	0	0	0	0
b	0	0	0	0	0	1	0	0	0	0	0	0
c	0	0	0	0	0	0	1	0	1	0	0	0
d	0	0	0	0	0	0	0	0	1	1	0	0
e	0	0	0	0	0	0	0	0	0	1	0	0
ab	0	0	0	0	0	0	0	0	0	0	0	0
ca	0	0	0	0	0	0	0	0	0	0	0	0
$bc+ca+ab$	0	0	0	0	0	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0	0	0	0	0	0
$eb+de+cd+ca+ab$	0	0	0	0	0	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0	0	0	0	0	0

Full matrix at termination. Observe:

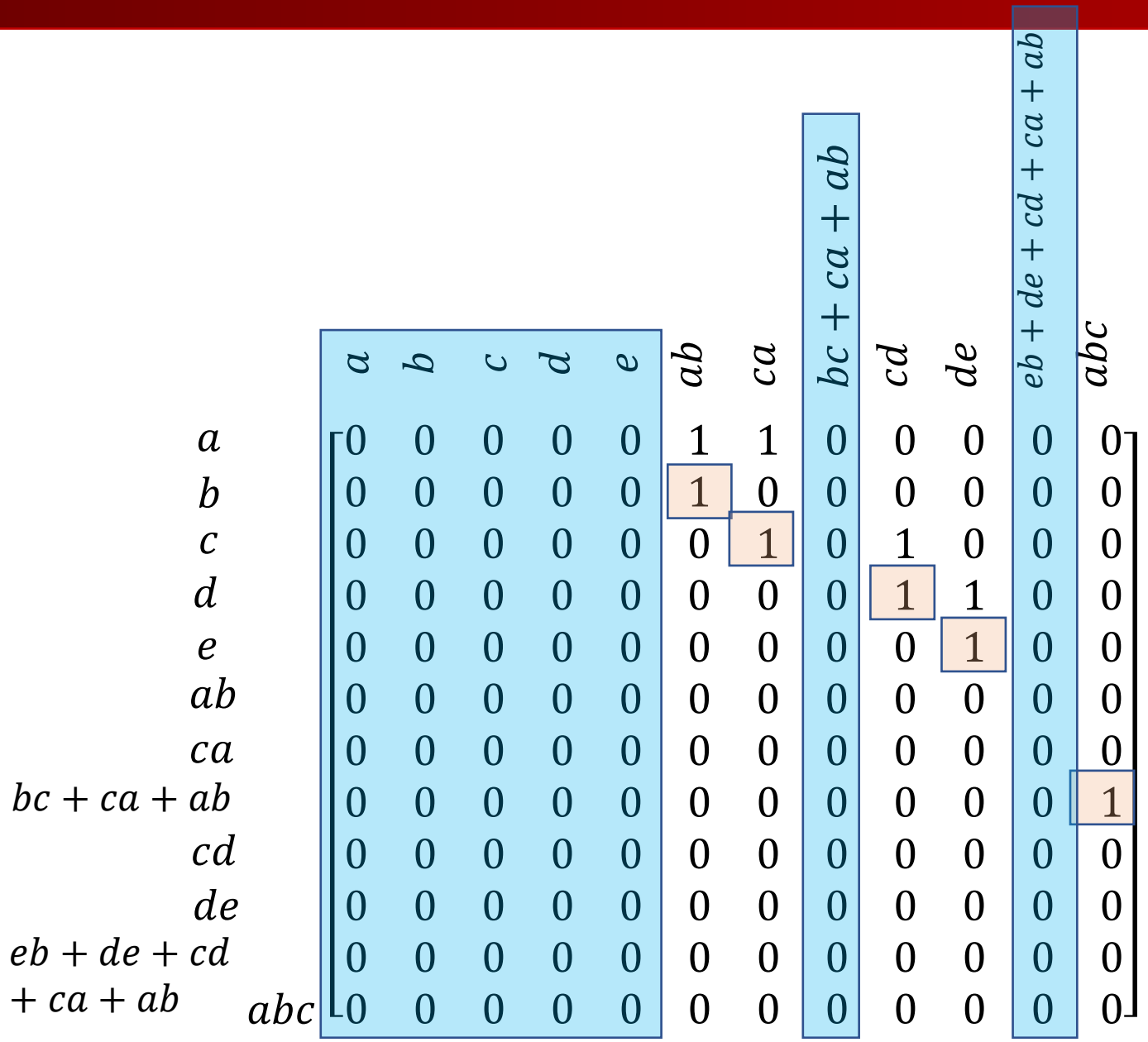
Reduction algorithm for homology

	a	b	c	d	e	ab	ca	$bc+ca+ab$	cd	de	$eb+de+cd+ca+ab$	abc
a	0	0	0	0	0	1	1	0	0	0	0	0
b	0	0	0	0	0	1	0	0	0	0	0	0
c	0	0	0	0	0	0	1	0	1	0	0	0
d	0	0	0	0	0	0	0	0	1	1	0	0
e	0	0	0	0	0	0	0	0	0	1	0	0
ab	0	0	0	0	0	0	0	0	0	0	0	0
ca	0	0	0	0	0	0	0	0	0	0	0	0
$bc+ca+ab$	0	0	0	0	0	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0	0	0	0	0	0
$eb+de+cd+ca+ab$	0	0	0	0	0	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0	0	0	0	0	0

Full matrix at termination. Observe:

- (Proposition) No two columns with the same low_R value

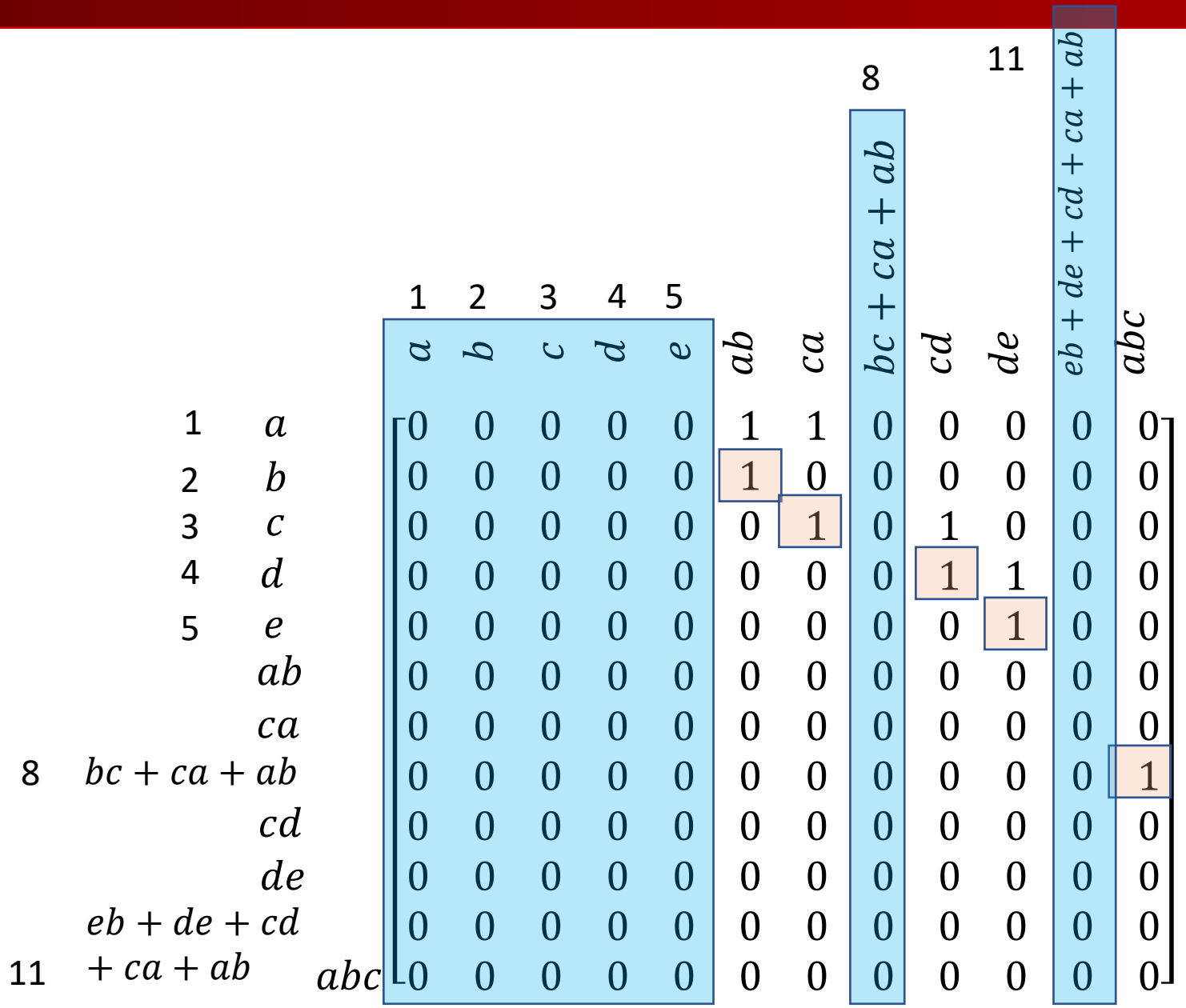
Reduction algorithm for homology



Full matrix at termination. Observe:

- (Proposition) No two columns with the same low_R value
- Zero columns can be read off immediately to get $\ker \partial_p$

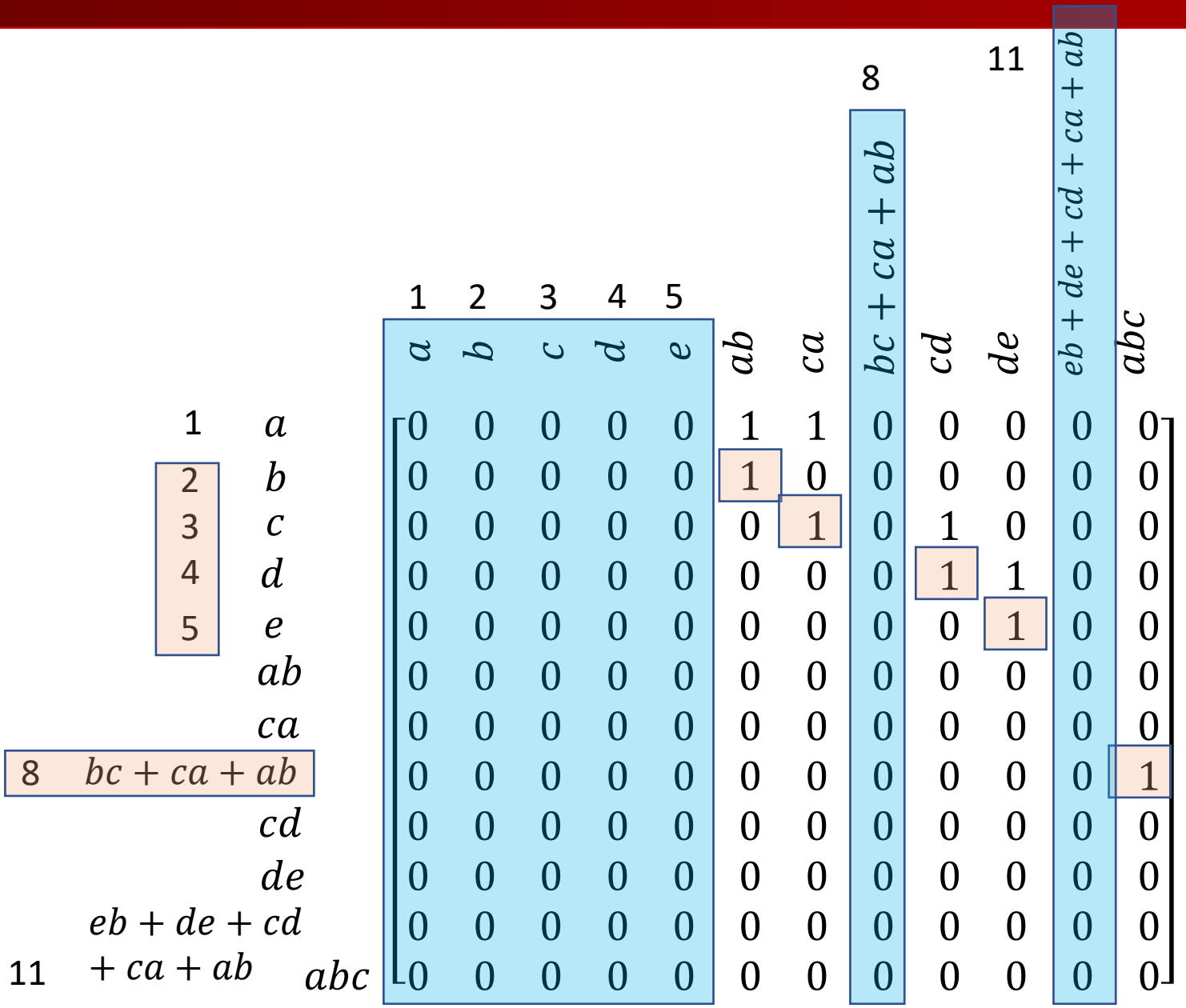
Reduction algorithm for homology



Full matrix at termination. Observe:

- (Proposition) No two columns with the same low_R value
- Zero columns can be read off immediately to get $\ker \partial_p$
- (Proposition) Indices i such that column i is zero but $i \neq low_R(j)$ for any j correspond exactly to the cycles that are not boundaries!
- Thus we get a basis for homology in each dimension.

Reduction algorithm for homology



Full matrix at termination. Observe:

- (Proposition) No two columns with the same low_R value
- Zero columns can be read off immediately to get $\ker \partial_p$
- (Proposition) Indices i such that column i is zero but $i \neq low_R(j)$ for any j correspond exactly to the cycles that are not boundaries!
- Thus we get a basis for homology in each dimension.

Reduction algorithm for homology

$H_0 \cong \mathbb{F}[a]$

$H_1 \cong \mathbb{F}[eb + de + cd + ca + ab]$

		1	2	3	4	5	6	7	8	9	10	11	12
		a	b	c	d	e	ab	ca	$bc+ca+ab$	cd	de	$eb+de+cd+ca+ab$	abc
1	a	0	0	0	0	0	1	1	0	0	0	0	0
2	b	0	0	0	0	0	1	0	0	0	0	0	0
3	c	0	0	0	0	0	0	1	0	1	0	0	0
4	d	0	0	0	0	0	0	0	0	1	1	0	0
5	e	0	0	0	0	0	0	0	0	0	1	0	0
	ab	0	0	0	0	0	0	0	0	0	0	0	0
	ca	0	0	0	0	0	0	0	0	0	0	0	0
8	$bc+ca+ab$	0	0	0	0	0	0	0	0	0	0	0	1
	cd	0	0	0	0	0	0	0	0	0	0	0	0
	de	0	0	0	0	0	0	0	0	0	0	0	0
	$eb+de+cd$	0	0	0	0	0	0	0	0	0	0	0	0
11	$+ca+ab$	0	0	0	0	0	0	0	0	0	0	0	0
	abc	0	0	0	0	0	0	0	0	0	0	0	0

Full matrix at termination. Observe:

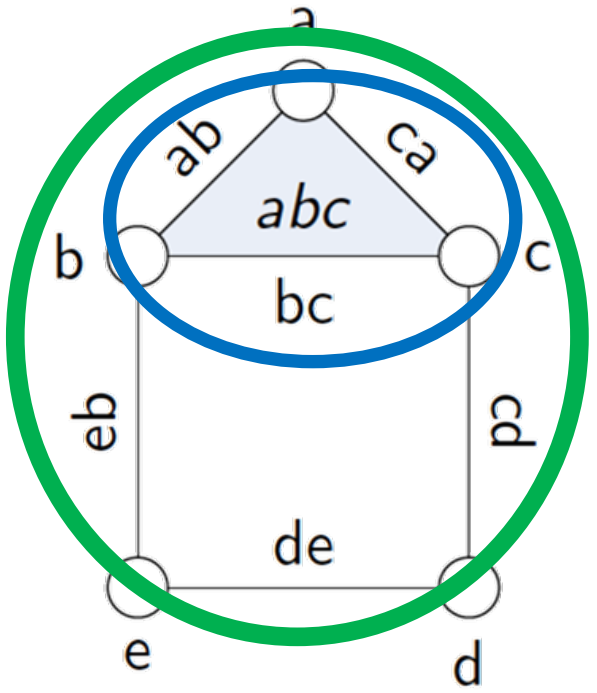
- (Proposition) No two columns with the same low_R value
- Zero columns can be read off immediately to get $\ker \partial_p$
- (Proposition) Indices i such that column i is zero but $i \neq low_R(j)$ for any j correspond exactly to the cycles that are not boundaries!
- Thus we get a basis for homology in each dimension.

Reduction algorithm for homology

$H_0 \cong \mathbb{F}[a]$

$H_1 \cong \mathbb{F}[eb + de + cd + ca + ab]$

		1	2	3	4	5	ab	ca	8 $bc + ca + ab$	cd	de	11 $eb + de + cd + ca + ab$	abc
		a	b	c	d	e							
1	a	0	0	0	0	0	1	1	0	0	0	0	0
2	b	0	0	0	0	0	1	0	0	0	0	0	0
3	c	0	0	0	0	0	0	1	0	1	0	0	0
4	d	0	0	0	0	0	0	0	0	1	1	0	0
5	e	0	0	0	0	0	0	0	0	0	1	0	0
	ab	0	0	0	0	0	0	0	0	0	0	0	0
	ca	0	0	0	0	0	0	0	0	0	0	0	0
8	$bc + ca + ab$	0	0	0	0	0	0	0	0	0	0	0	1
	cd	0	0	0	0	0	0	0	0	0	0	0	0
	de	0	0	0	0	0	0	0	0	0	0	0	0
	$eb + de + cd$	0	0	0	0	0	0	0	0	0	0	0	0
11	$+ ca + ab$	0	0	0	0	0	0	0	0	0	0	0	0
	abc												

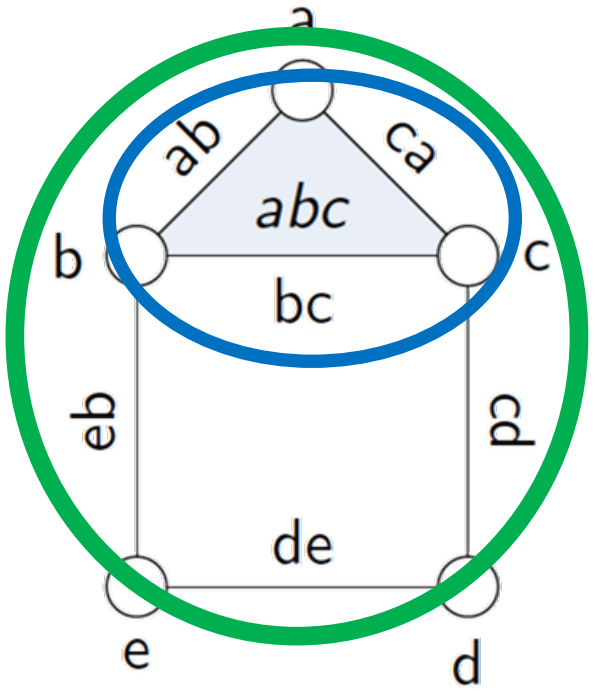


The bases we deserve, not the bases we need

$H_0 \cong \mathbb{F}[a]$

$H_1 \cong \mathbb{F}[eb + de + cd + ca + ab]$

		1	2	3	4	5	ab	ca	8 $bc + ca + ab$	cd	de	11 $eb + de + cd + ca + ab$	abc
		a	b	c	d	e							
1	a	0	0	0	0	0	1	1	0	0	0	0	0
2	b	0	0	0	0	0	1	0	0	0	0	0	0
3	c	0	0	0	0	0	0	1	0	1	0	0	0
4	d	0	0	0	0	0	0	0	0	1	1	0	0
5	e	0	0	0	0	0	0	0	0	0	1	0	0
	ab	0	0	0	0	0	0	0	0	0	0	0	0
	ca	0	0	0	0	0	0	0	0	0	0	0	0
8	$bc + ca + ab$	0	0	0	0	0	0	0	0	0	0	0	1
	cd	0	0	0	0	0	0	0	0	0	0	0	0
	de	0	0	0	0	0	0	0	0	0	0	0	0
11	$eb + de + cd + ca + ab$	0	0	0	0	0	0	0	0	0	0	0	0
abc		0	0	0	0	0	0	0	0	0	0	0	0



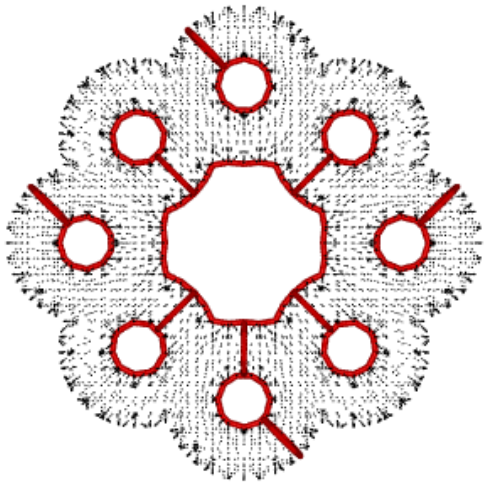
Caveat: homology involves an equivalence class, so we get "representative cycles". E.g. we got $eb + de + cd + cca + ab$ instead of $eb + de + cd + bc$.

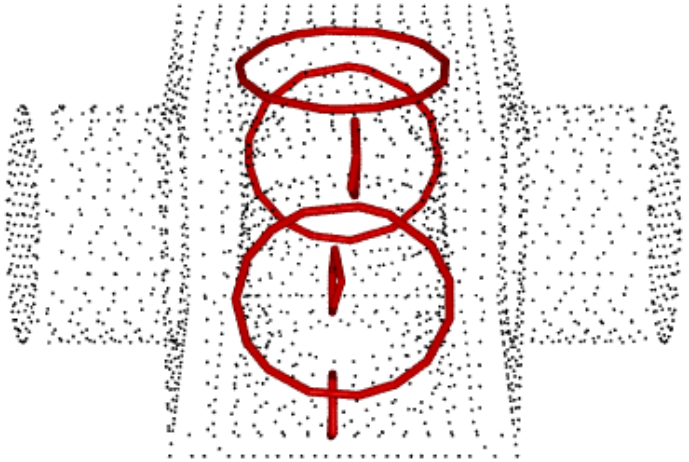
Operating at the algebraic level gives up some control over geometric correspondence.

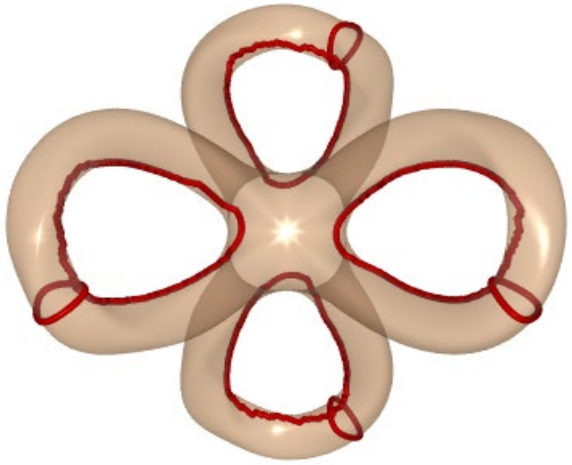
ShortLoop

Oleksiy Busaryev, Tamal Dey, Jian Sun, Yusu Wang

Figures from http://web.cse.ohio-state.edu/~dey.8/shortloop-pictures.html







Persistent homology

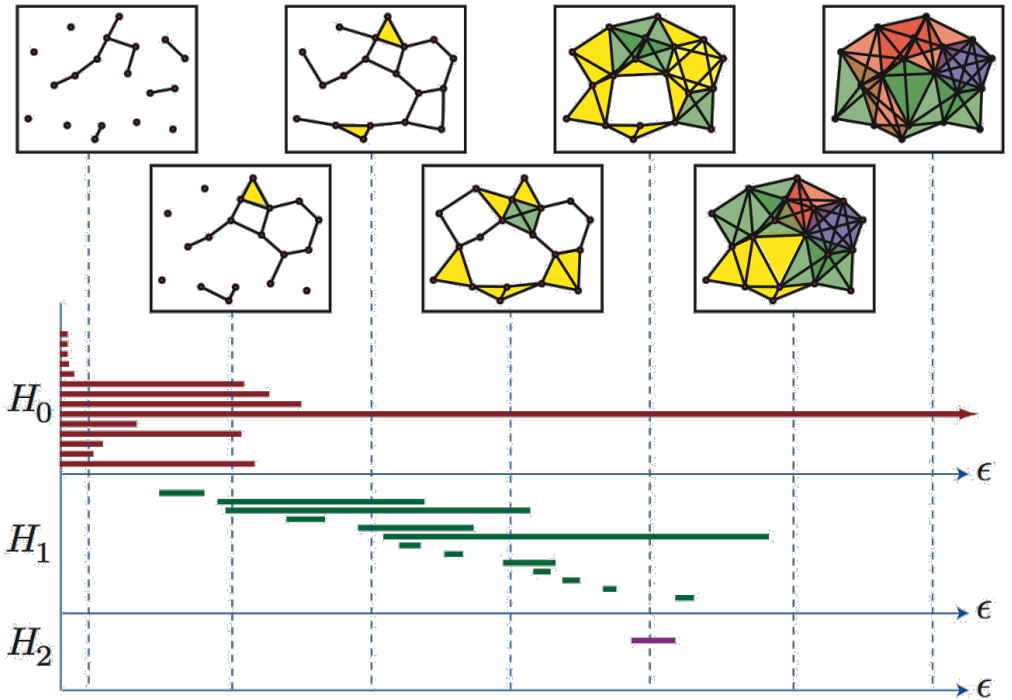
Pipeline

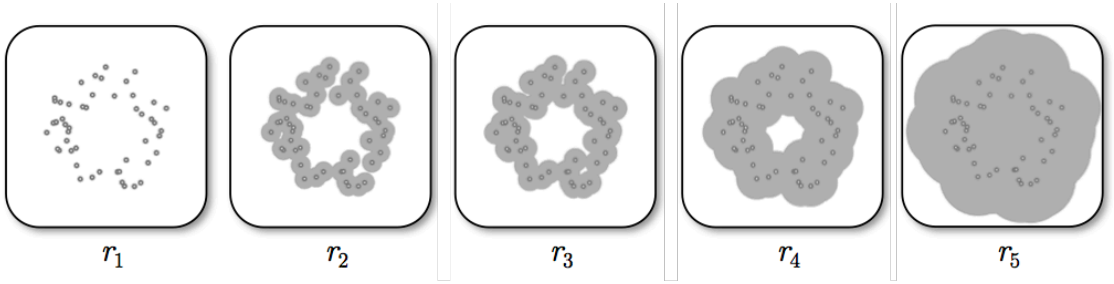
Dataset:
- **Euclidean point cloud data**
- **Metric spaces**
- **Triangulated shape**
- Graphs, matrices…

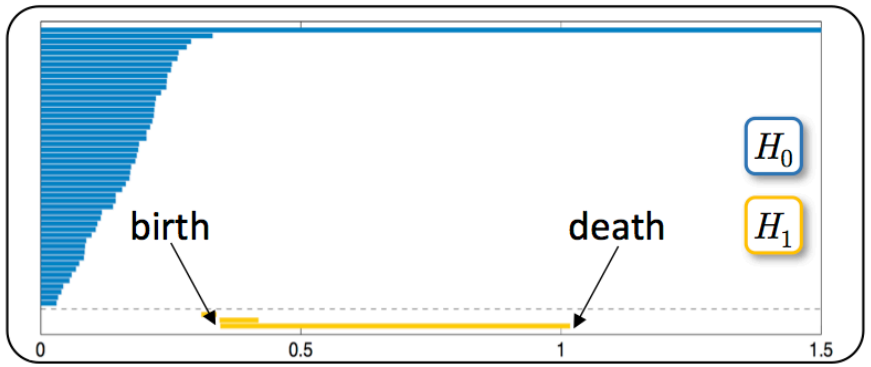
Filtered simplicial complex
- Possible to have both inclusions and deletions (dynamic data)

Vector spaces with linear maps: a **persistent vector space**

Visual summary: a **persistence diagram** or **persistence barcode**
- Persistence landscapes for ML-friendly output







Reduction algorithm for persistent homology

Ulrich Bauer, *Ripser: efficient computation of Vietoris–Rips persistence barcodes*

Persistence algorithm:

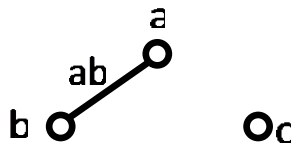
- When setting up boundary matrices/overall incidence matrix, order basis vectors according to order of appearance of simplices
- Apply reduction algorithm from before
- Read off persistence pairs $\{(i,j) \mid i = low_R(j) \neq 0\}$ and essential indices $\{i \mid R[:,i] = 0,\ i \neq low_R(j) \text{ for any } j\}$
- Persistence pairs track representative cycles that are born at index i and die at index j
- Essential indices track cycles that are born at index i and live forever
- Complexity: Gaussian elimination, so essentially cubic in the number of simplices (state-of-the-art: matrix multiplication time)

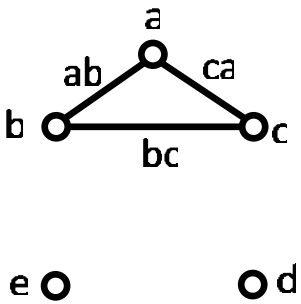
Back to the house example

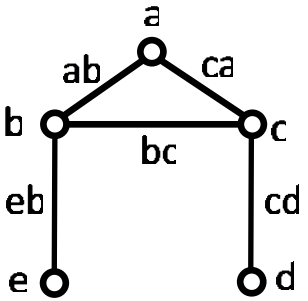
Entry time: 1 1 1 1 1 1 2 2 3 3 4 5

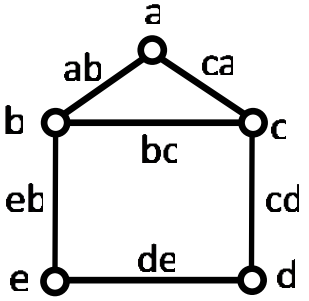
Index: 1 2 3 4 5 6 7 8 9 10 11 12

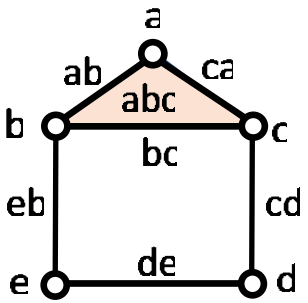
	a	b	c	d	e	ab	ca	bc	cd	eb	de	abc
a	0	0	0	0	0	1	1	0	0	0	0	0
b	0	0	0	0	0	1	0	1	0	1	0	0
c	0	0	0	0	0	0	1	1	1	0	0	0
d	0	0	0	0	0	0	0	0	1	0	1	0
e	0	0	0	0	0	0	0	0	0	1	1	0
ab	0	0	0	0	0	0	0	0	0	0	0	1
ca	0	0	0	0	0	0	0	0	0	0	0	1
bc	0	0	0	0	0	0	0	0	0	0	0	1
cd	0	0	0	0	0	0	0	0	0	0	0	0
eb	0	0	0	0	0	0	0	0	0	0	0	0
de	0	0	0	0	0	0	0	0	0	0	0	0
abc	0	0	0	0	0	0	0	0	0	0	0	0





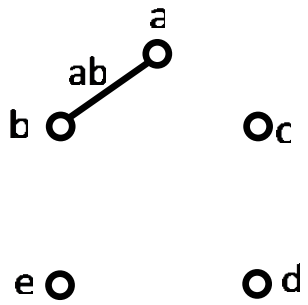


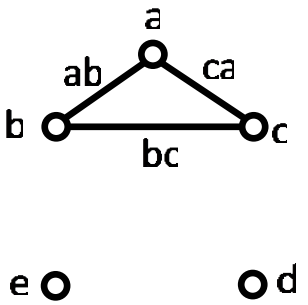


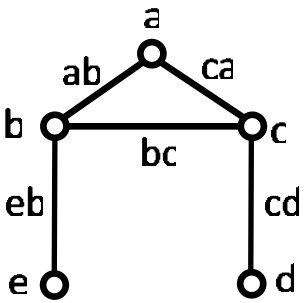


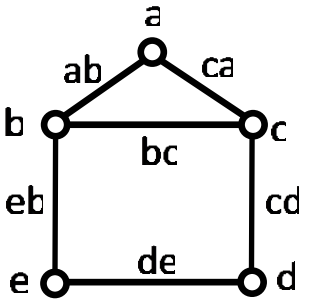
Back to the house example

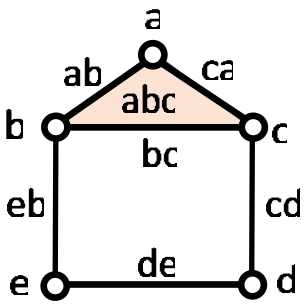
		1	2	3	4	5	ab	ca	8 $bc+ca+ab$	cd	eb	11 $de+eb+cd+ca+ab$	abc
		a	b	c	d	e							
1	a	0	0	0	0	0	1	1	0	0	0	0	0
2	b	0	0	0	0	0	1	0	0	0	1	0	0
3	c	0	0	0	0	0	0	1	0	1	0	0	0
4	d	0	0	0	0	0	0	0	0	1	0	0	0
5	e	0	0	0	0	0	0	0	0	0	1	0	0
	ab	0	0	0	0	0	0	0	0	0	0	0	0
	ca	0	0	0	0	0	0	0	0	0	0	0	0
8	$bc+ca+ab$	0	0	0	0	0	0	0	0	0	0	0	1
	cd	0	0	0	0	0	0	0	0	0	0	0	0
	eb	0	0	0	0	0	0	0	0	0	0	0	0
11	$de+eb+cd+ca+ab$	0	0	0	0	0	0	0	0	0	0	0	0
	abc	0	0	0	0	0	0	0	0	0	0	0	0











Back to the house example

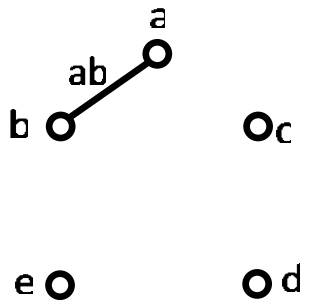
Persistence pairs: $\{(2,6),(3,7),(4,9),(5,10),(8,12)\}$
(column indices)

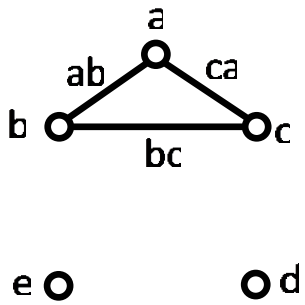
Lifetimes (dim 0): $\{[1,1),[1,2),[1,3),[1,3)\}$
(time indices)

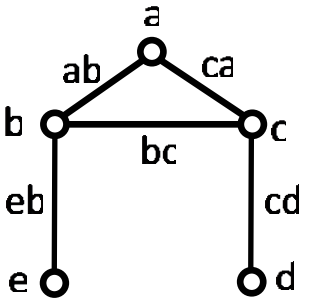
Lifetimes (dim 1): $\{[2,5)\}$
(time indices)

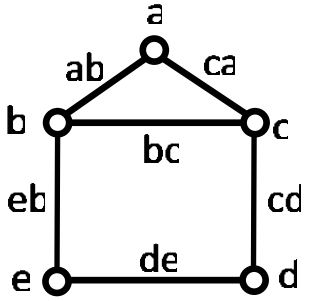
Essential indices: $\{1,11\}$
(column indices)

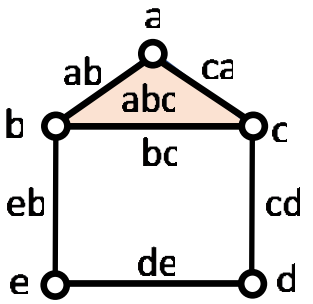
Lifetimes: $\{[1,\infty)$ (connected component), $[4,\infty)$ (loop)$\}$
(time indices)











Simplices are “creators” or “destroyers”

- Every simplex either creates a homology class or destroys one
- Specifically, a k-simplex either creates a k-dimensional feature or destroys a $(k-1)$-dimensional feature

A deeper look at the algorithm

- Why the restriction on directions of column operations?
- Why is persistent homology different from computing standard homology a bunch of times in parallel?

$$0 \rightarrow H_p\left(\Sigma^{(1)}\right) \rightarrow H_p(\Sigma^{(2)}) \rightarrow \ldots \rightarrow H_p(\Sigma^{(n-1)}) \rightarrow H_p(\Sigma^{(n)})$$

A deeper look at the algorithm

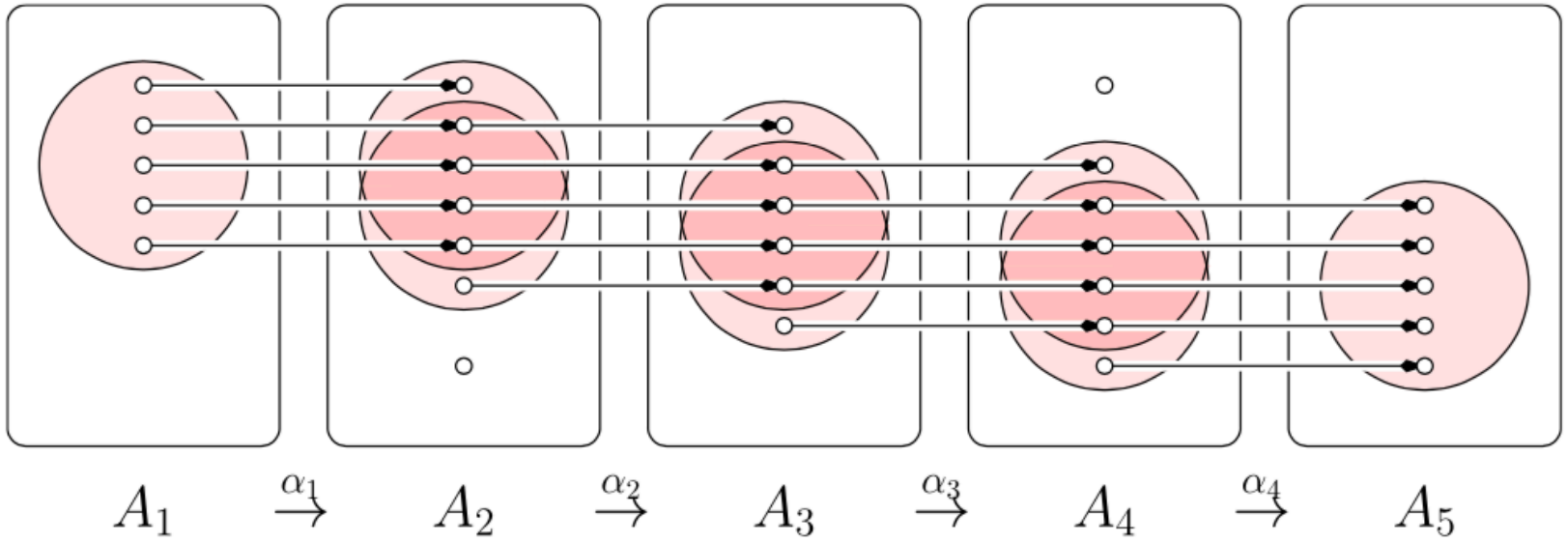
- Why the restriction on directions of column operations?
- Why is persistent homology different from computing standard homology a bunch of times in parallel?

$$0 \to H_p\left(\Sigma^{(1)}\right) \to H_p\left(\Sigma^{(2)}\right) \to \ldots \to H_p\left(\Sigma^{(n-1)}\right) \to H_p\left(\Sigma^{(n)}\right)$$

Obtaining bases for the individual vector spaces is not sufficient. To encode linear maps, we need compatible bases.

A deeper look at the algorithm

Figure from Edelsbrunner, Jablonski, Mrozek, *The persistent homology of a self-map*



- Left-to-right ordering of chain complex basis vectors + left-to-right column operations gets us to compatible bases for homology
- "Existence and uniqueness" comes from commutative algebra

PH as applied representation theory

Carlsson, de Silva, *Zigzag Persistence*

- $\mathbb{V} := V_1 \xrightarrow{f_1} V_2 \xrightarrow{f_2} \ldots \to V_{n-1} \xrightarrow{f_{n-1}} V_n$ = $\cdot \to \cdot \to$ ... $\to \cdot \to \cdot$ ✚ choice of vector spaces and linear maps

- This is a quiver representation. Given equal-length quiver reps $\mathbb{W}, \mathbb{X}$, say $\mathbb{V} = \mathbb{W} \oplus \mathbb{X}$ if $V_i = W_i \oplus X_i$ and $f_i(W_i) \subseteq W_{i+1}$, $f_i(X_i) \subseteq X_{i+1}$ for each i

Theorem (Gabriel '72): Any $\mathbb{V}$ as above can be written $\mathbb{V} = \bigoplus_i \mathbb{I}(b_i, d_i)$, where

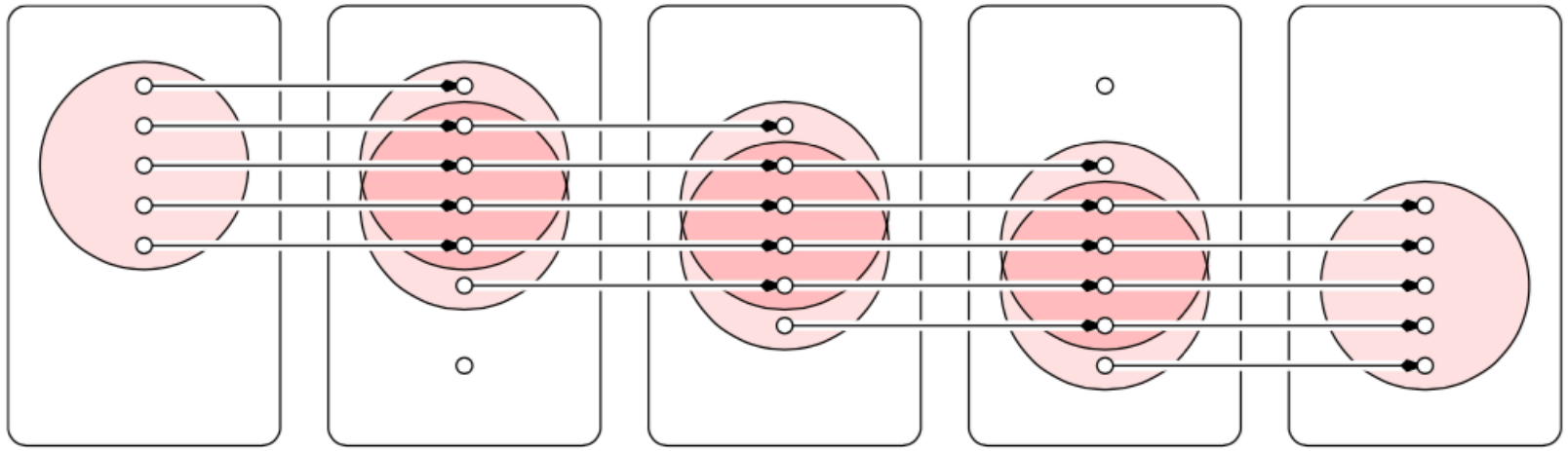
$$\mathbb{I}(b_i, d_i) = 0 \to 0 \to \cdots 0 \to \mathbb{F} \to \mathbb{F} \to \mathbb{F} \to \cdots \to \mathbb{F} \to 0 \to 0 \to 0$$

Index 1 | Index b_i | Index d_i | Index n

Theorem (Krull-Remak-Schmidt): Such a decomposition is unique up to relabeling.

PH as applied representation theory

Carlsson, de Silva, *Zigzag Persistence*



Theorem (Gabriel '72): Any $\mathbb{V}$ as above can be written $\mathbb{V} = \bigoplus_i \mathbb{I}(b_i, d_i)$, where

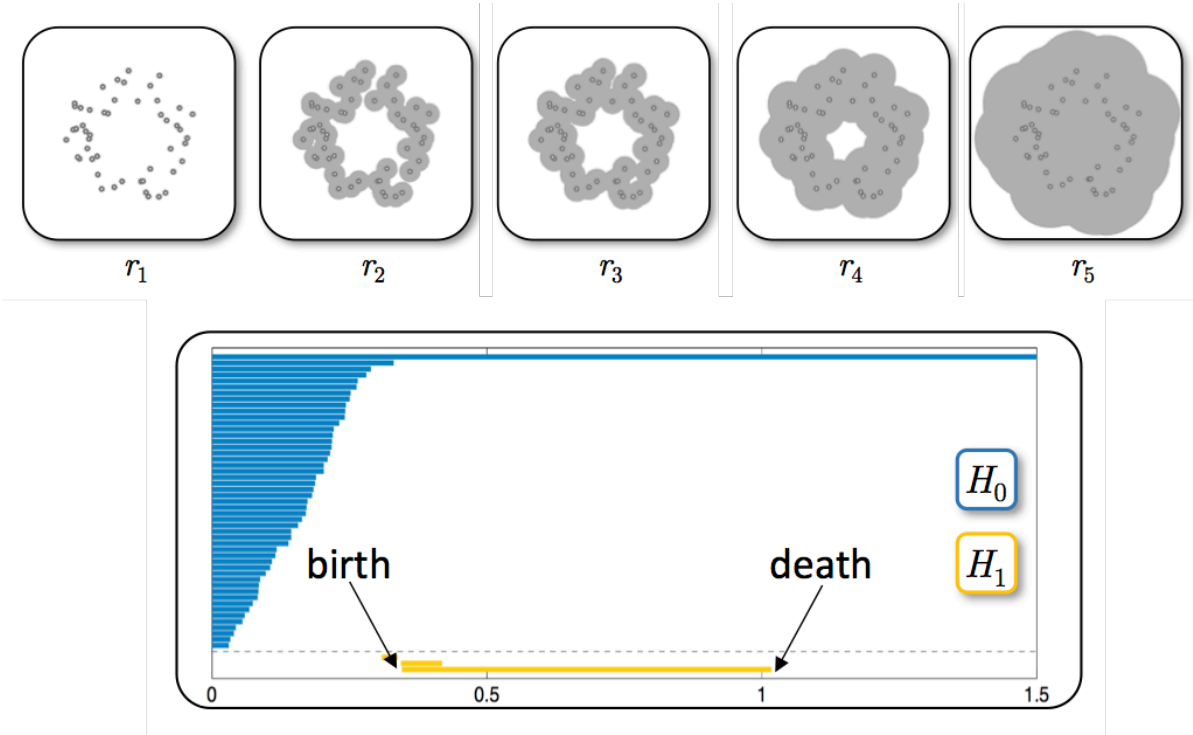
$$\mathbb{I}(b_i, d_i) = 0 \to 0 \to \cdots 0 \to \mathbb{F} \to \mathbb{F} \to \mathbb{F} \to \cdots \to \mathbb{F} \to 0 \to 0 \to 0$$

Index 1 | Index b_i | Index d_i | Index n

Theorem (Krull-Remak-Schmidt): Such a decomposition is unique up to relabeling.

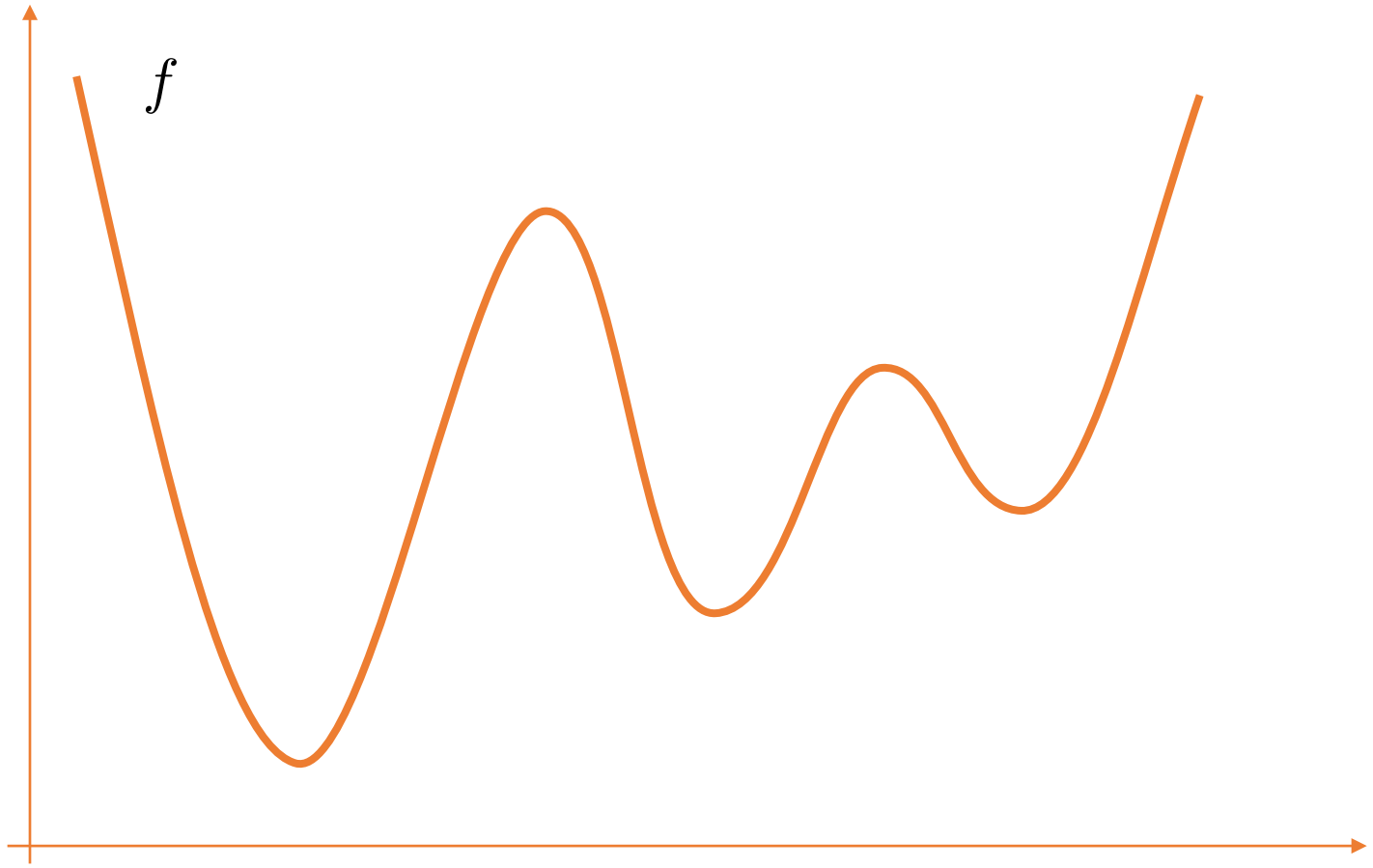
Sublevel Set Persistence

- For general point cloud or metric data, we build filtered simplicial complexes using the metric as a filter

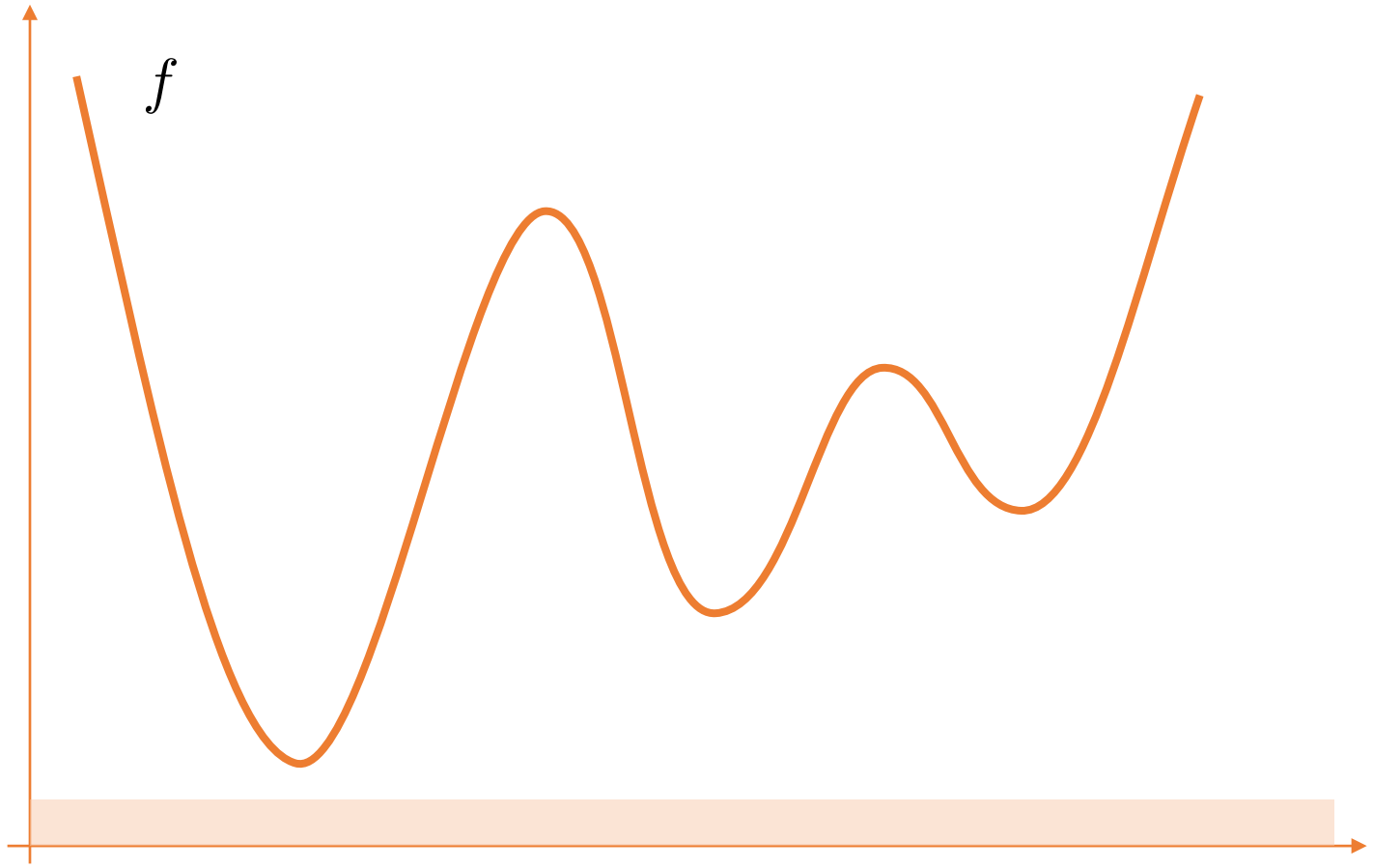


- When given a function $f: \mathcal{M} \to \mathbb{R}$ where $\mathcal{M}$ is a triangulated space (i.e. a simplicial complex at the outset), we can use f as a filter

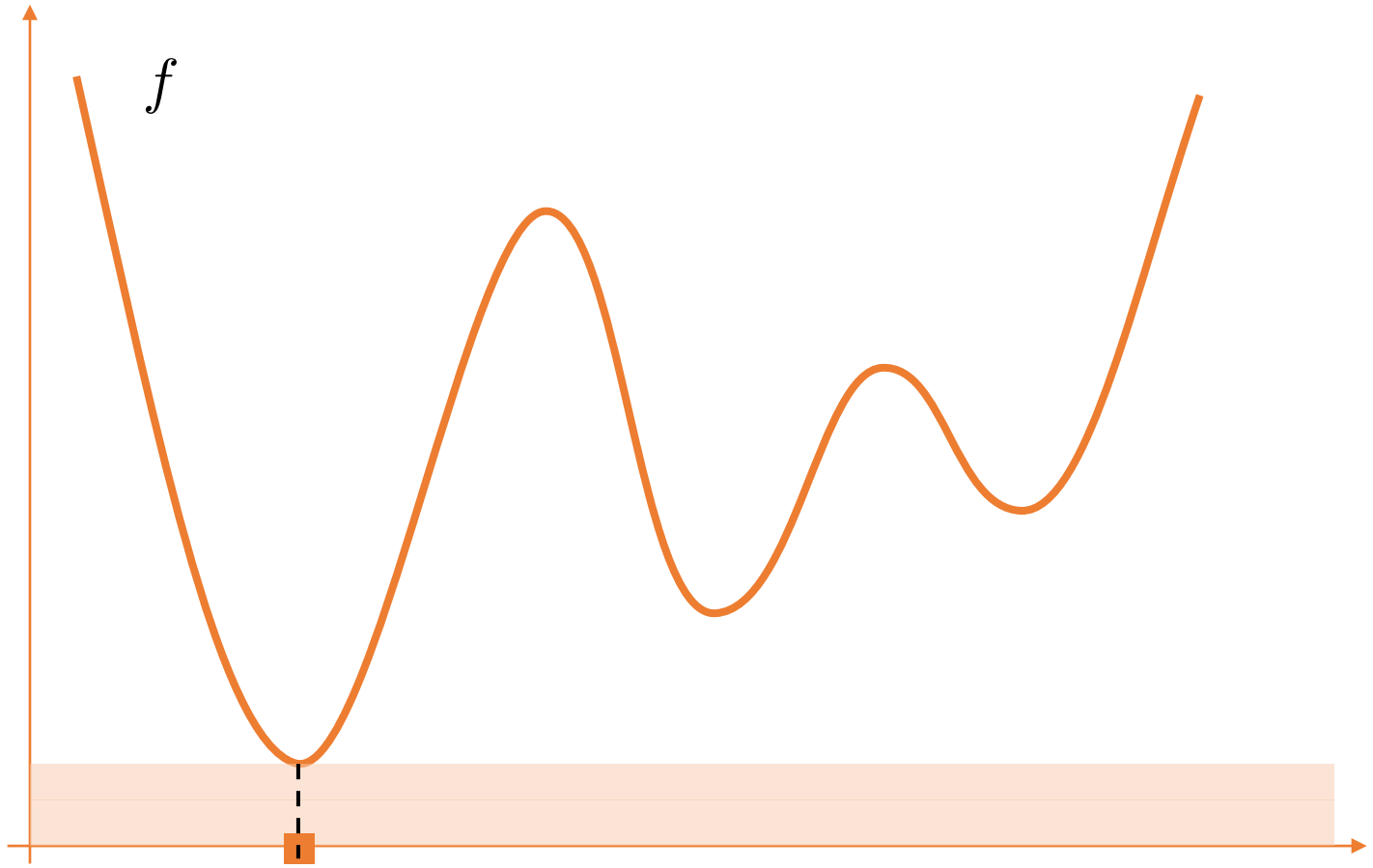
Sublevel Set Persistence



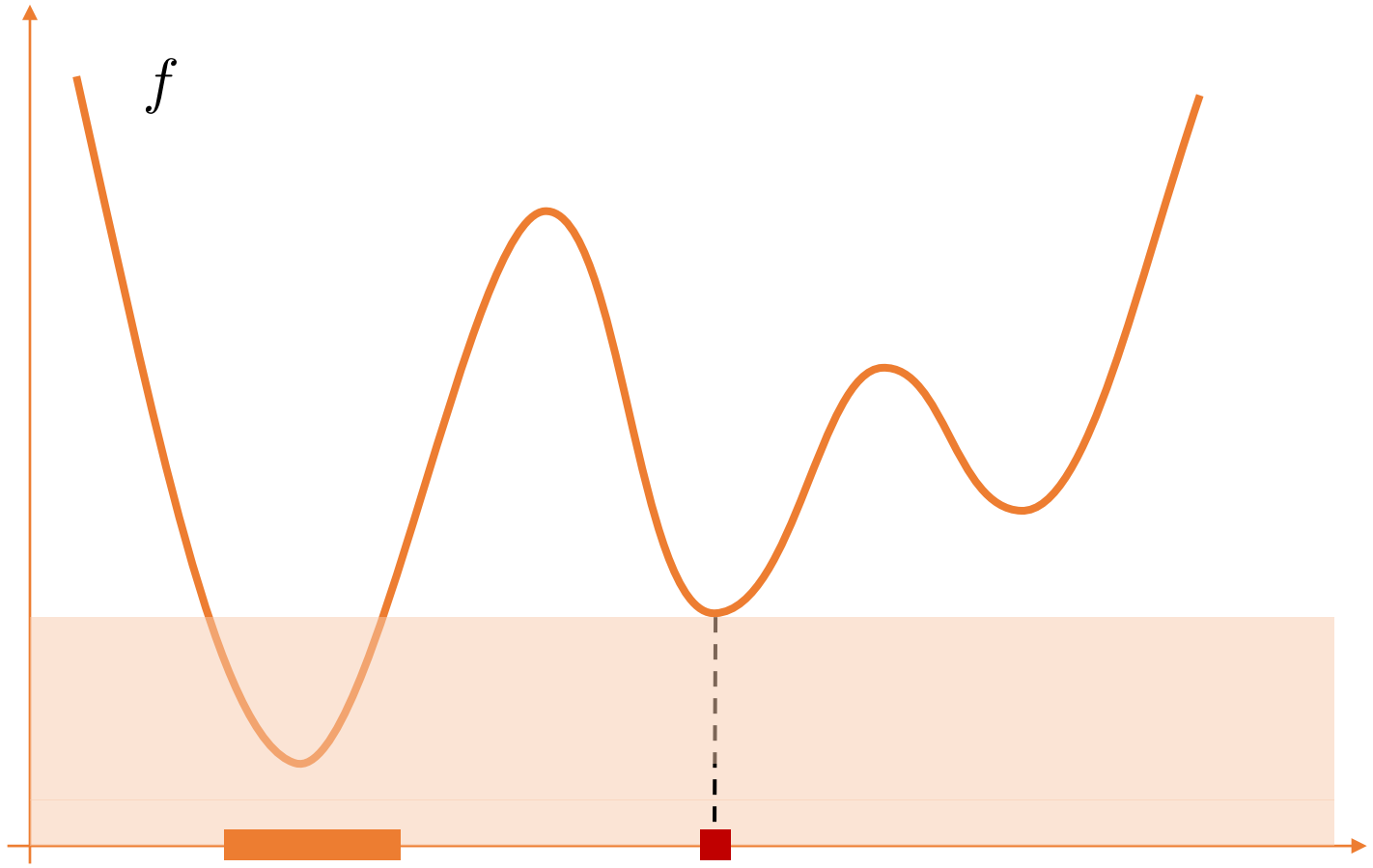
Sublevel Set Persistence



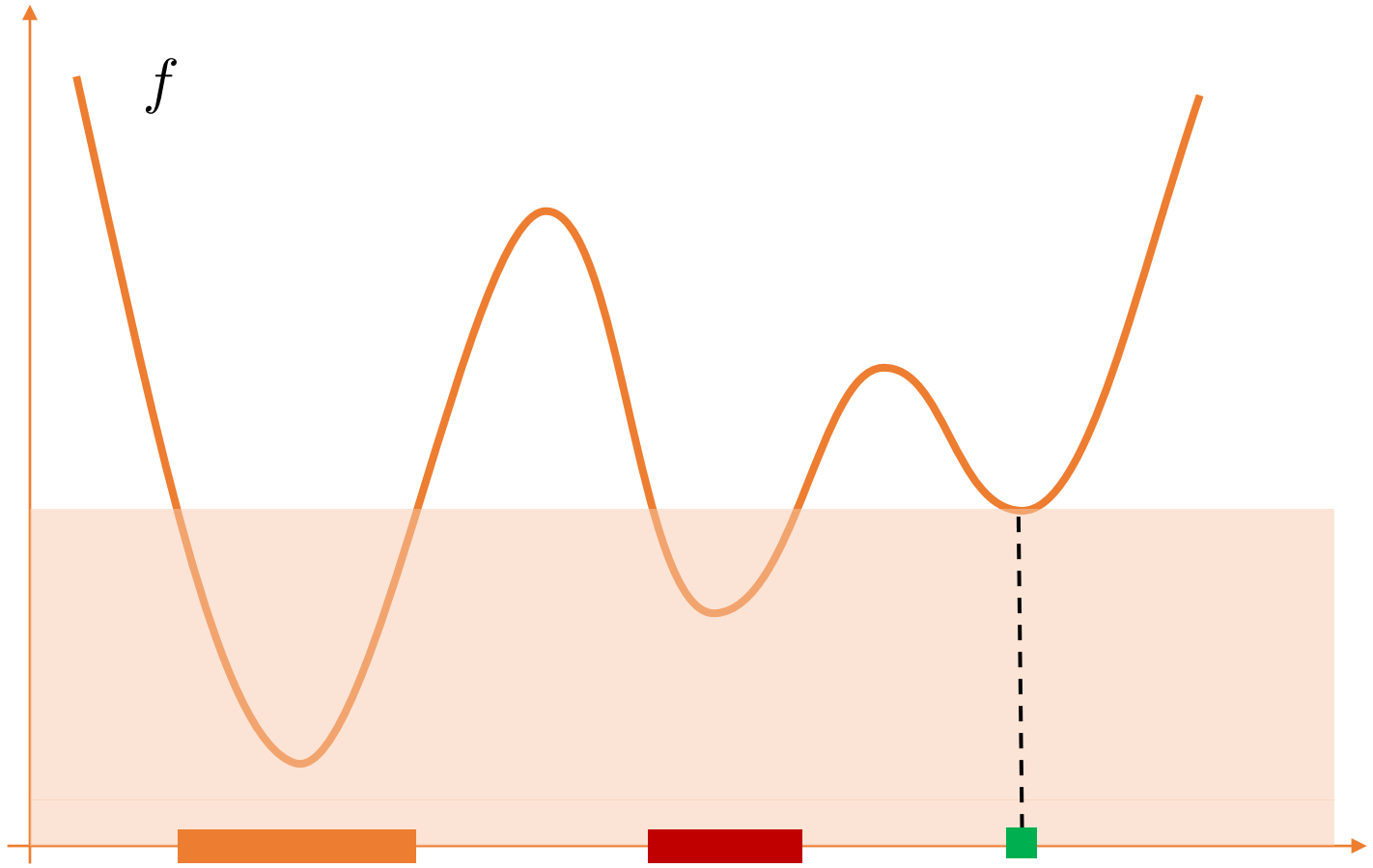
Sublevel Set Persistence



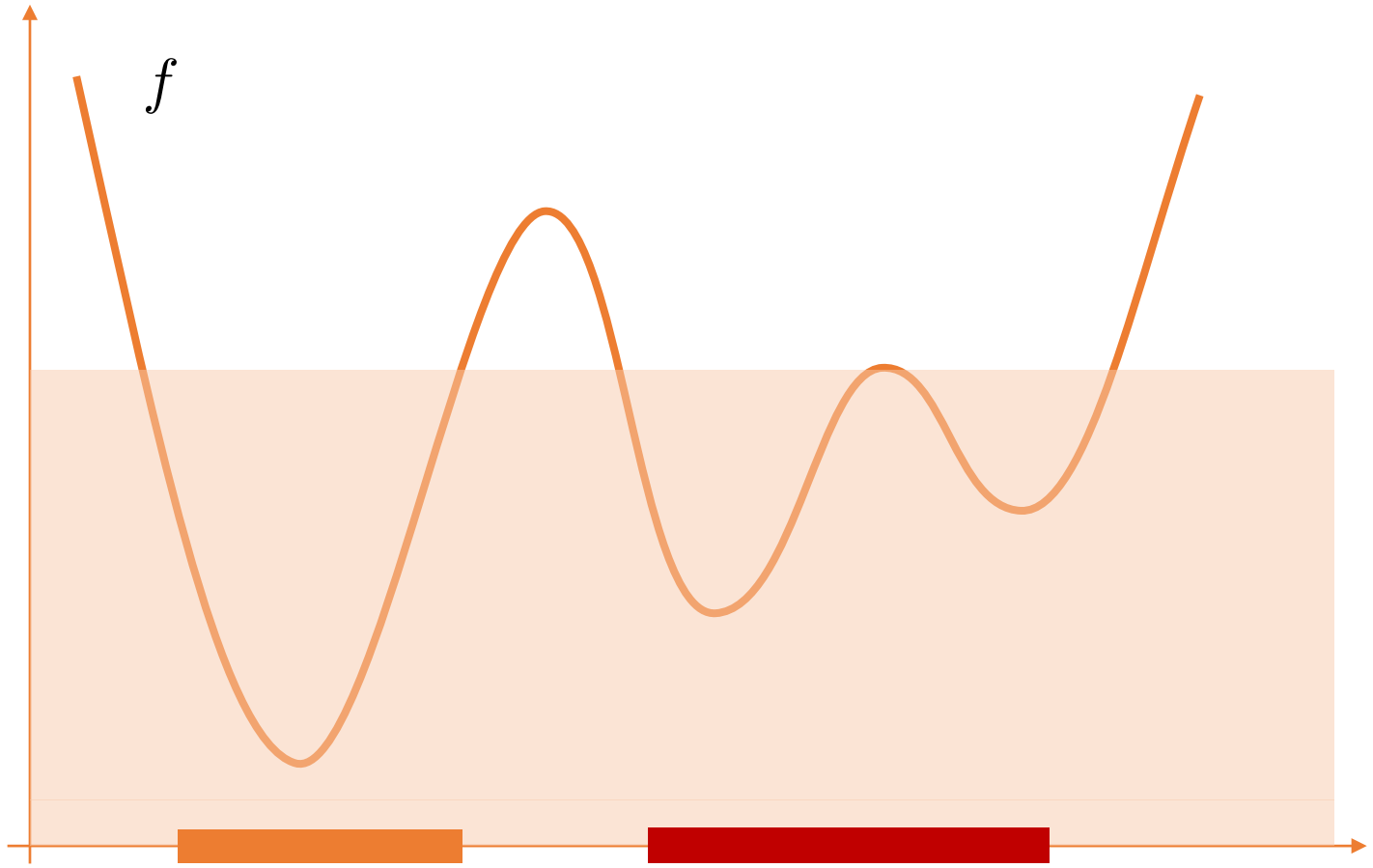
Sublevel Set Persistence



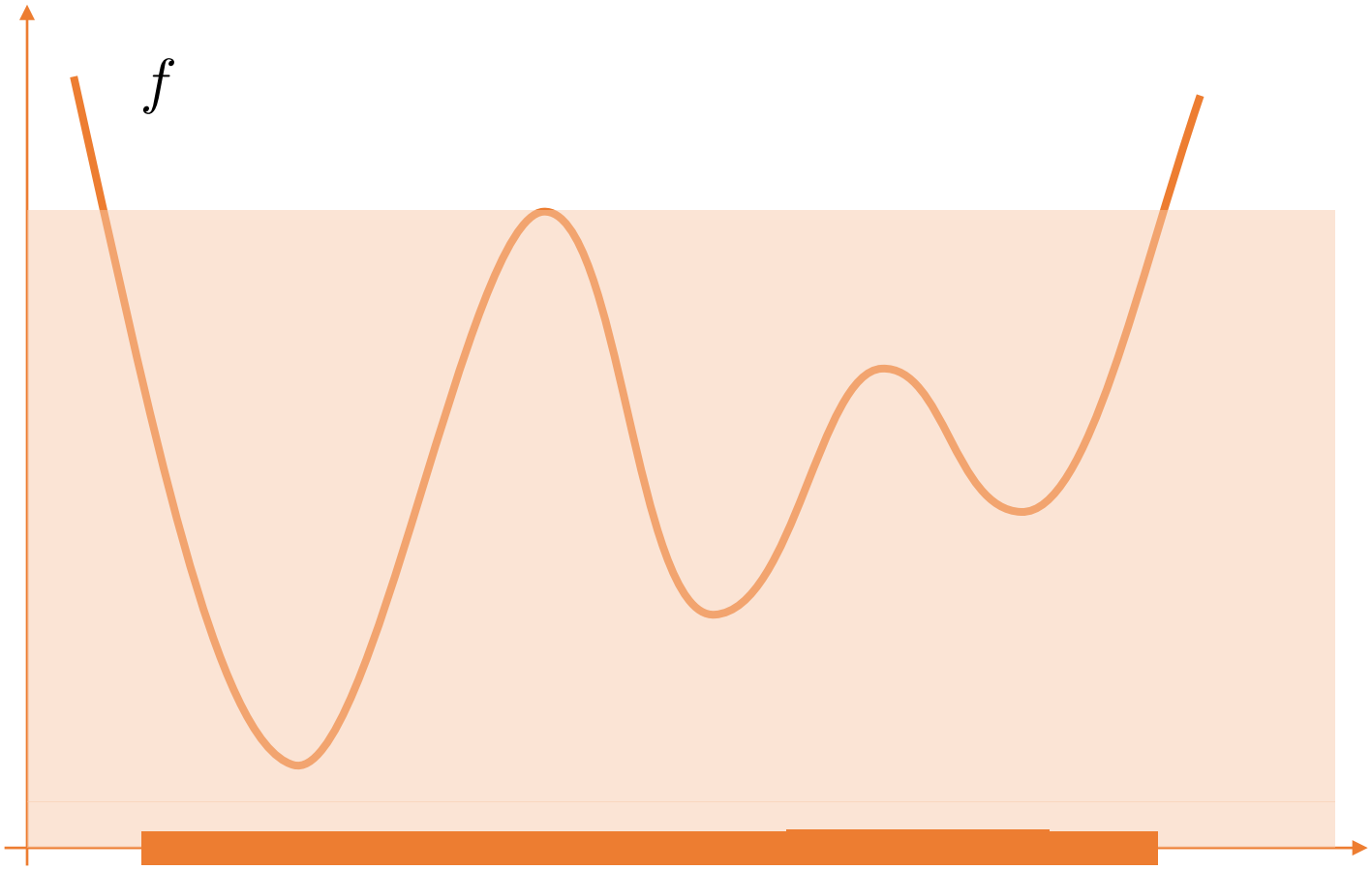
Sublevel Set Persistence



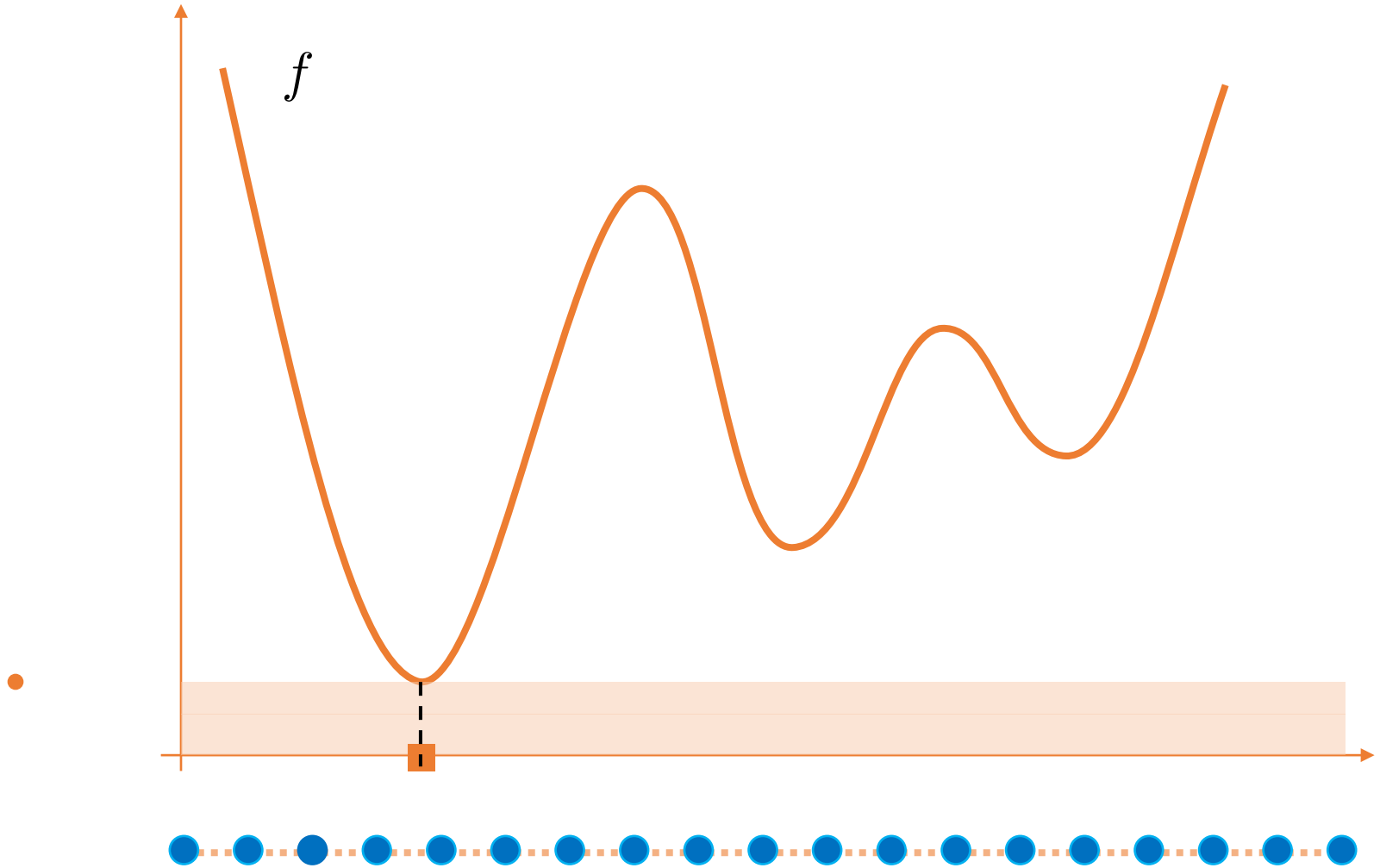
Sublevel Set Persistence



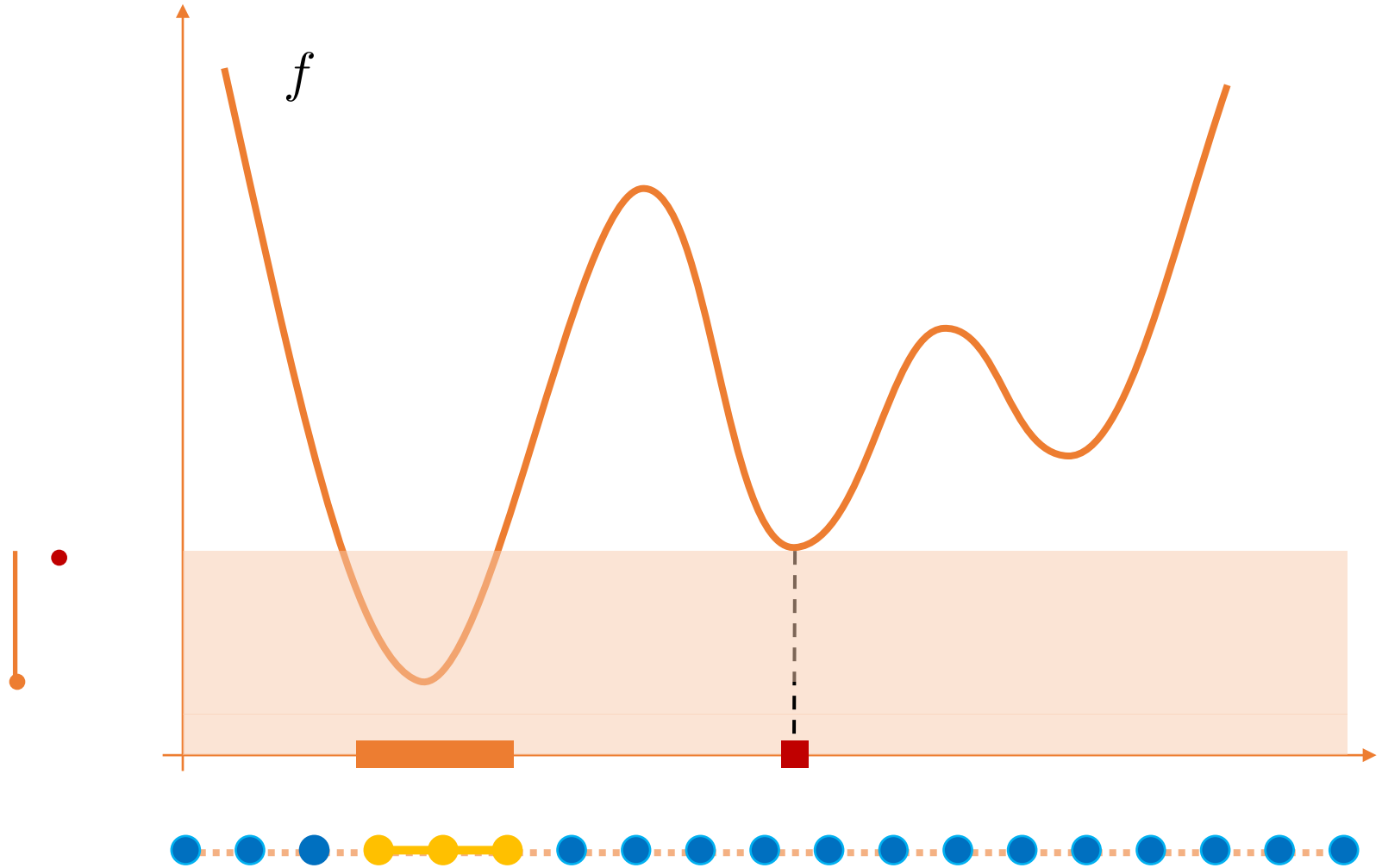
Sublevel Set Persistence



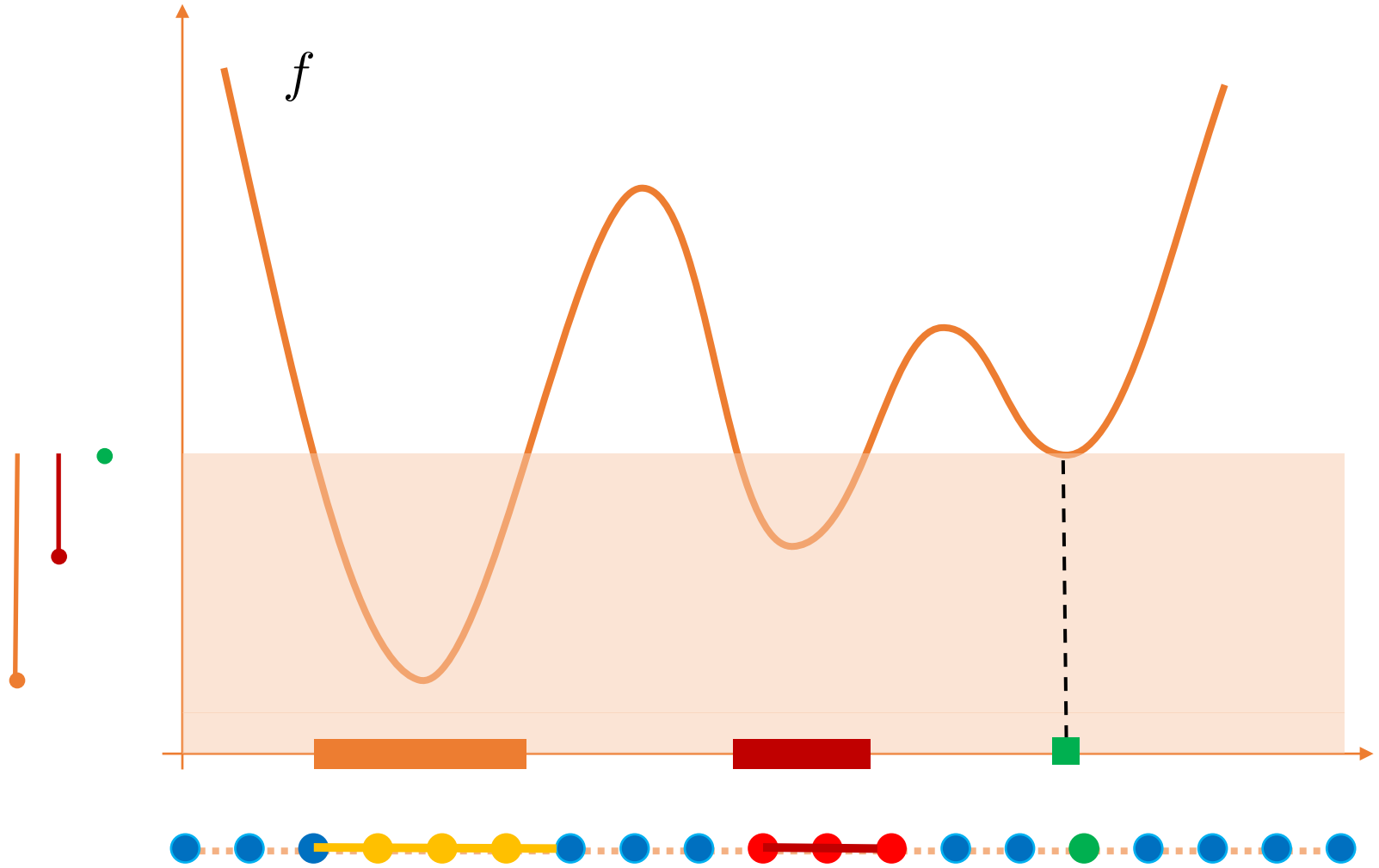
Sublevel Set Persistence



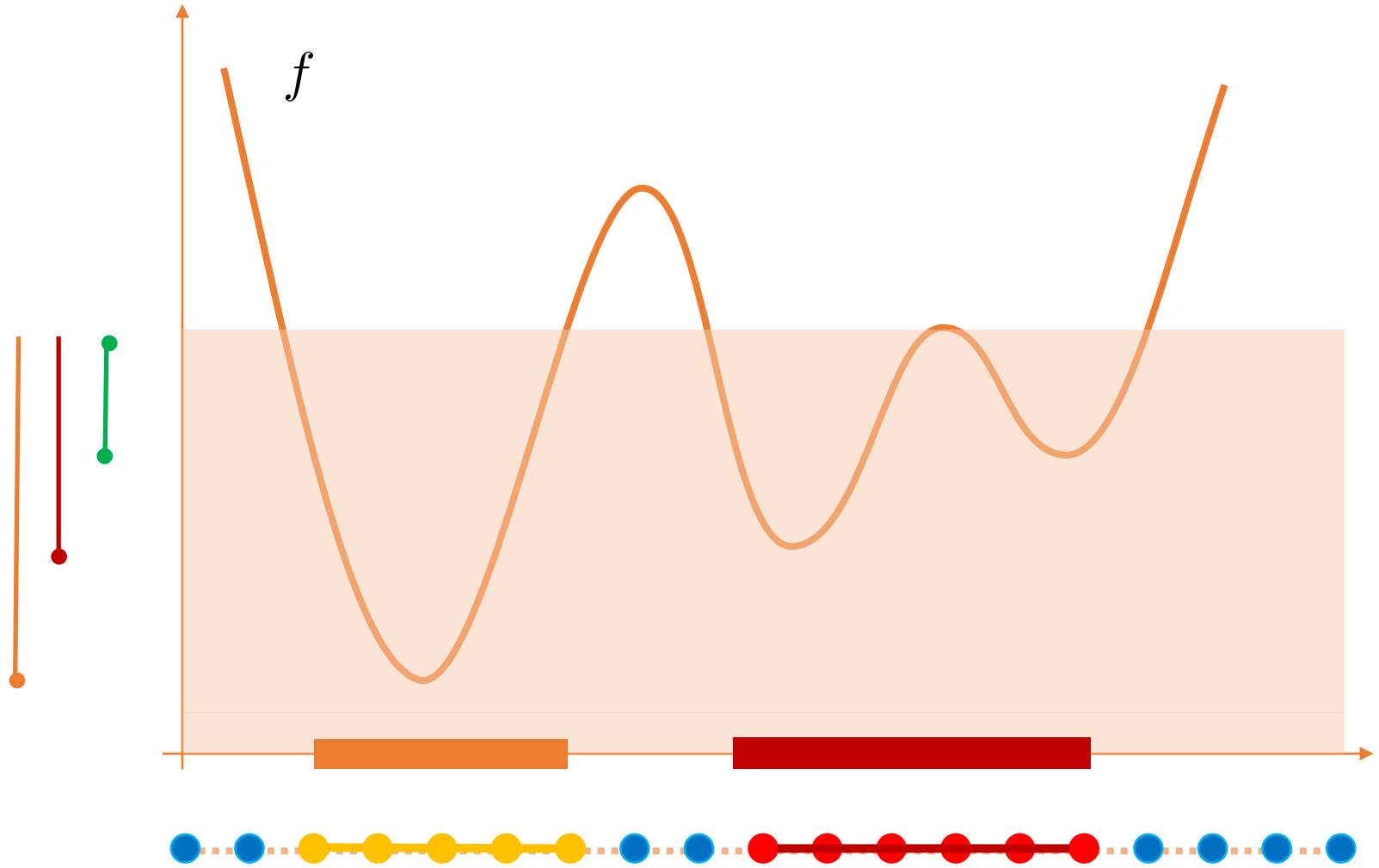
Sublevel Set Persistence



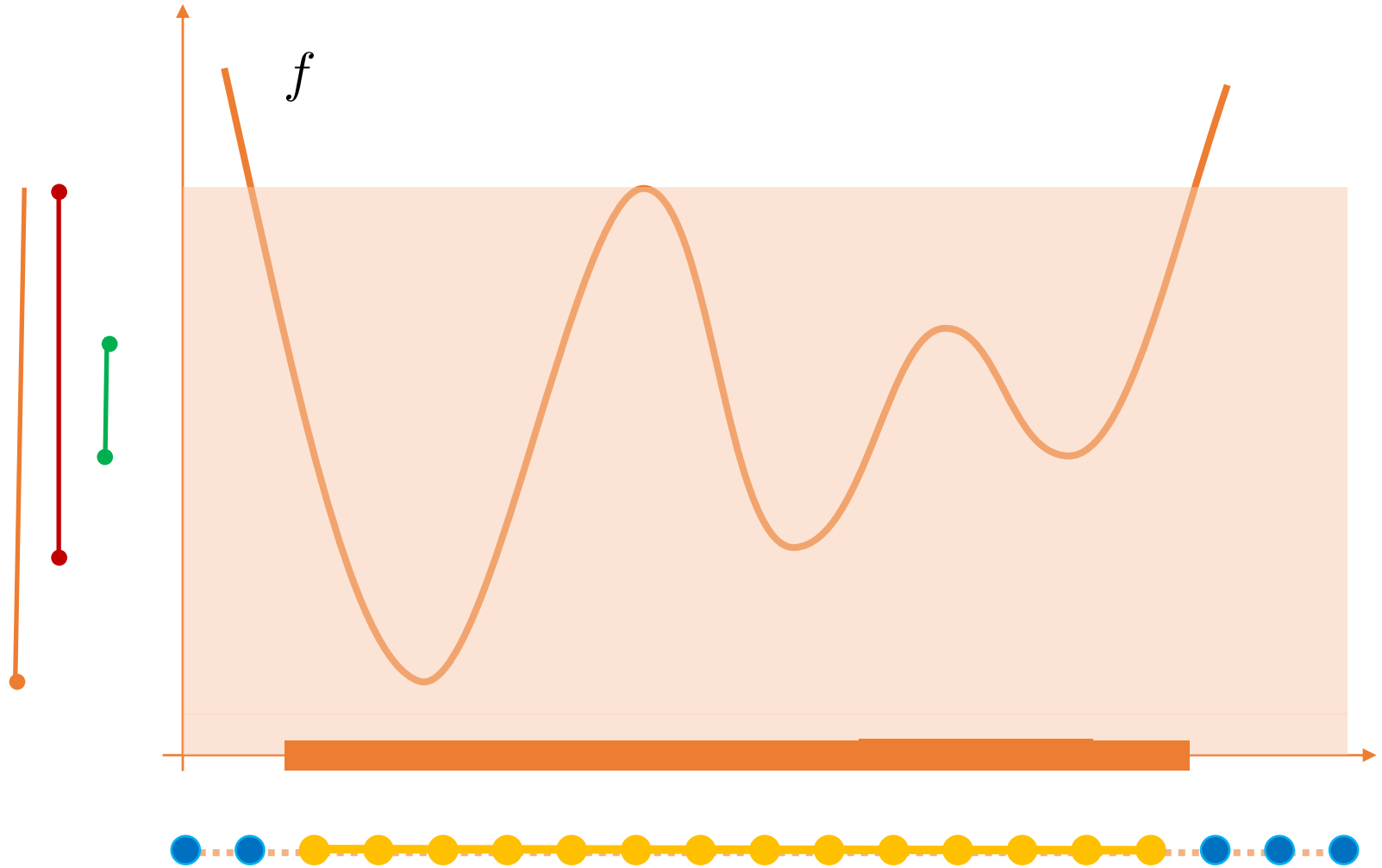
Sublevel Set Persistence



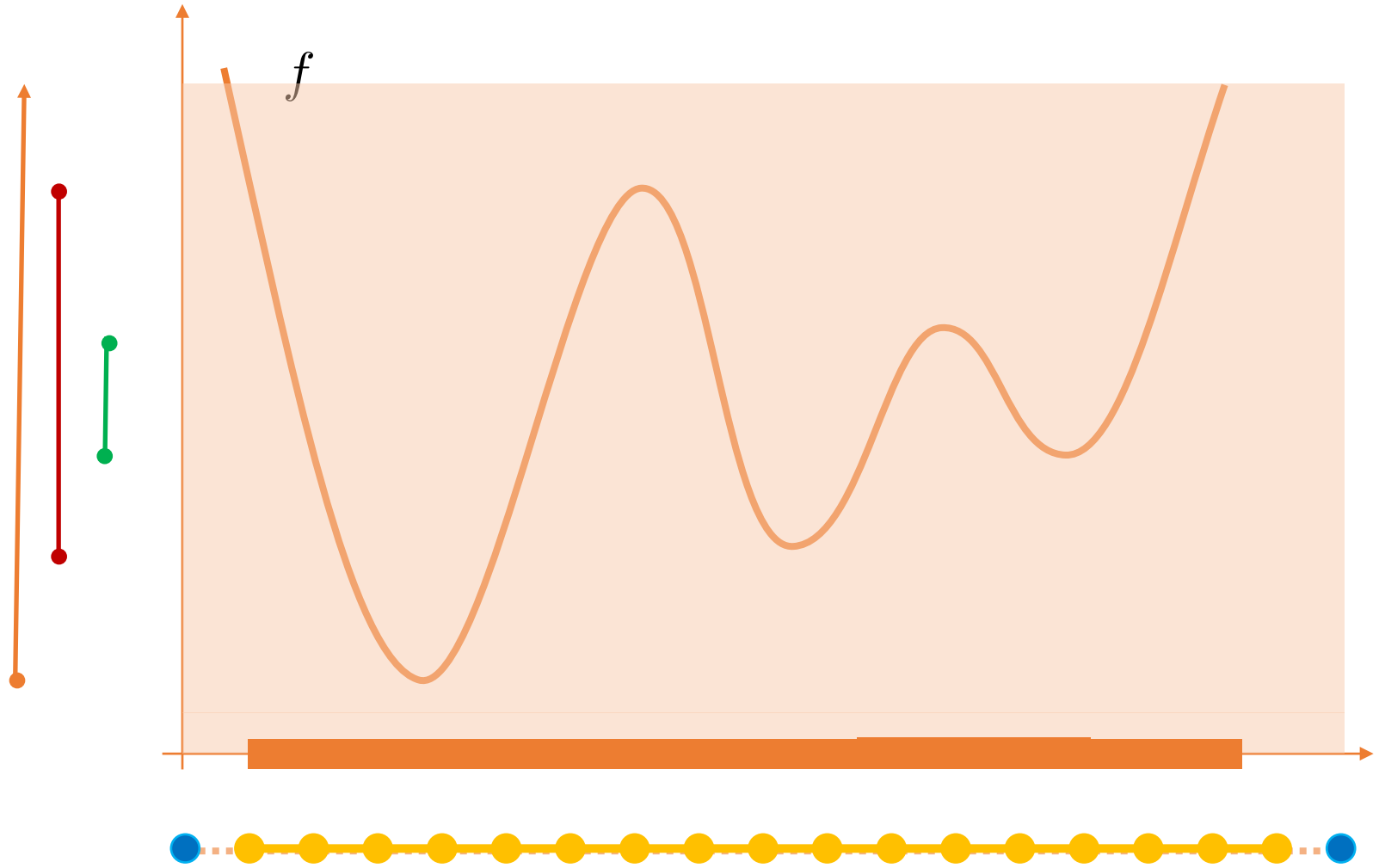
Sublevel Set Persistence



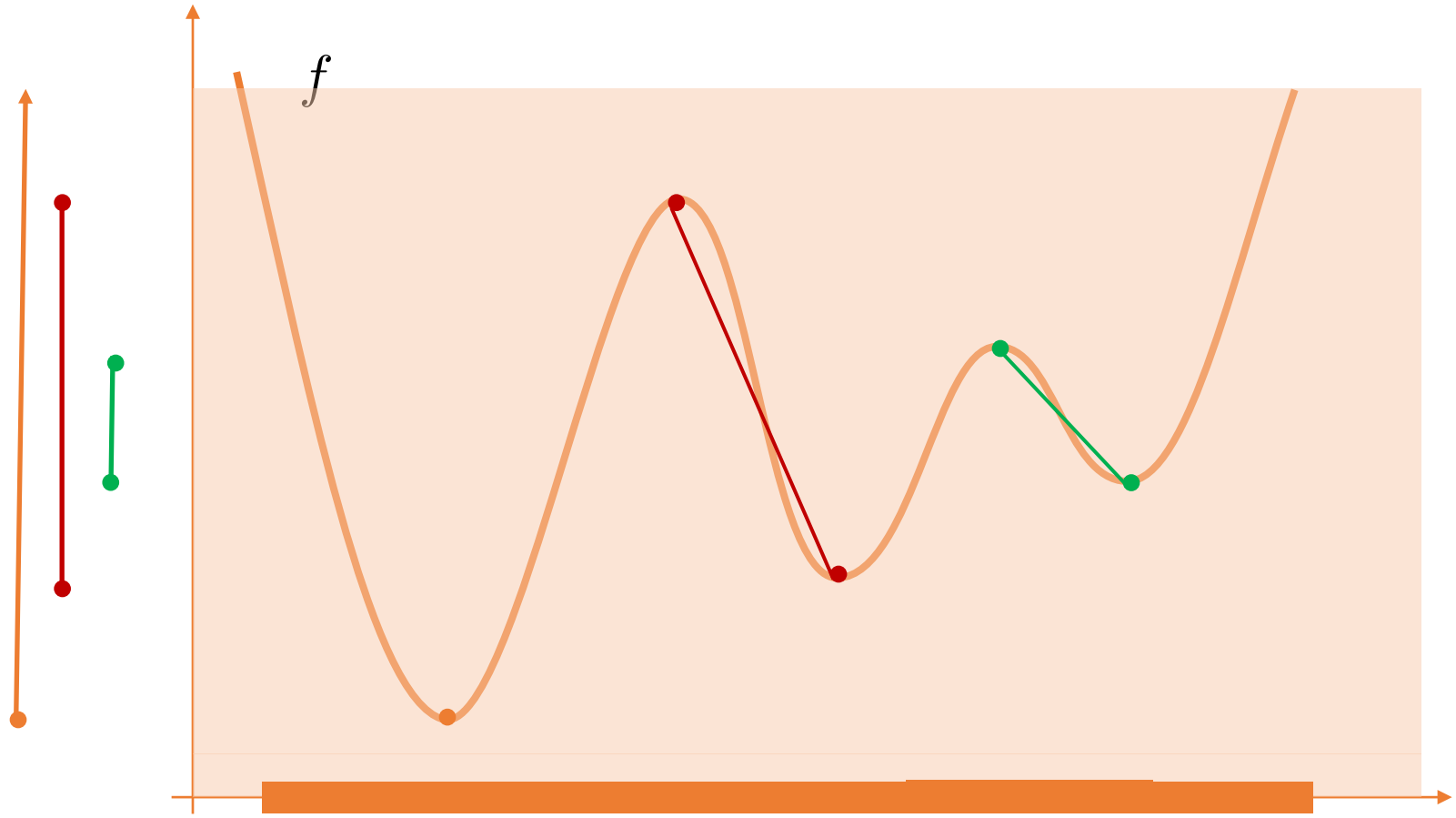
Sublevel Set Persistence



Persistence Gives a Pairing



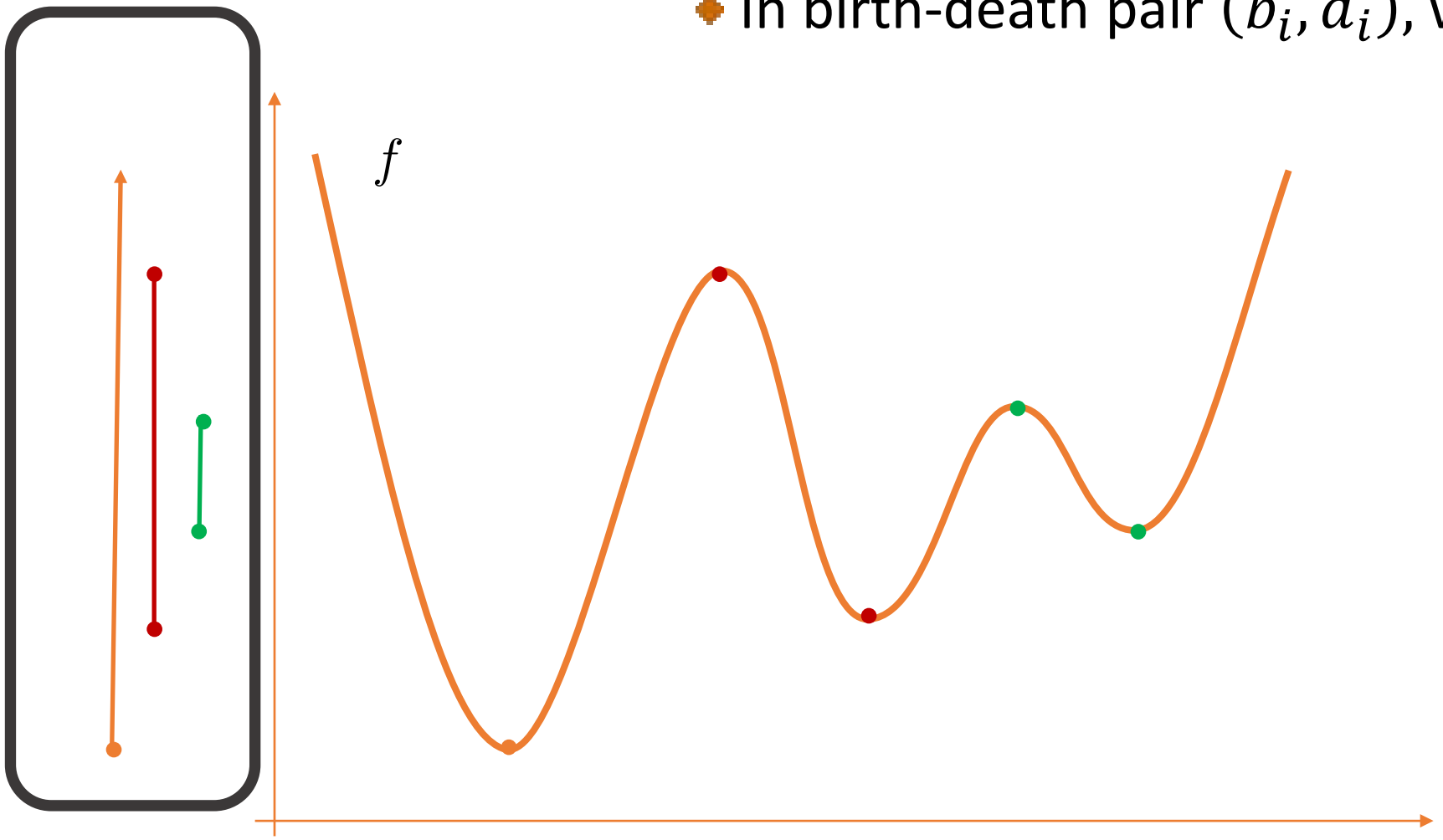
Elder Rule



Persistence barcodes and diagrams

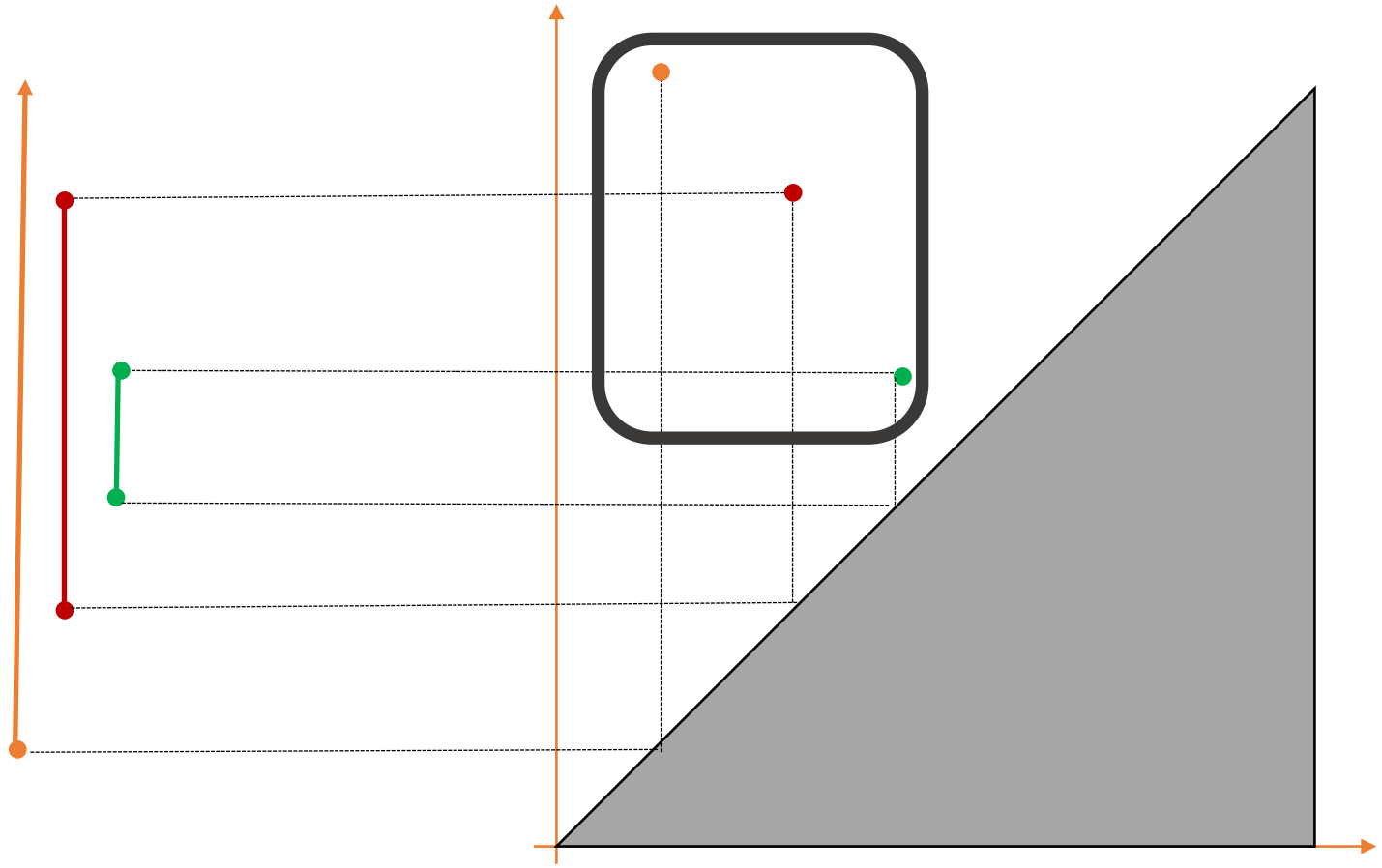
Persistence pairing yields a barcode

- In birth-death pair (b_i, d_i), we have $d_i \geq b_i$



Equivalently, a diagram

- Pairs (b_i, d_i) are represented as points on upper-half plane
- Often (but not always), short bars correspond to "topological noise"



Stability of persistent homology

- Persistent homology outputs are robust to small perturbations of data

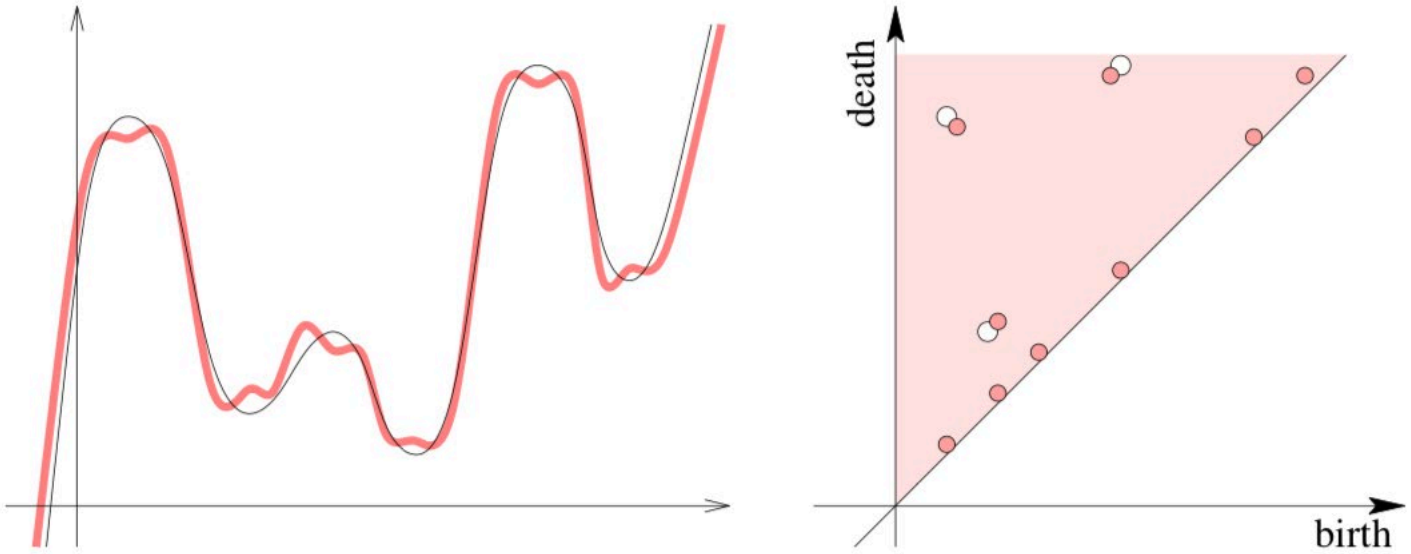


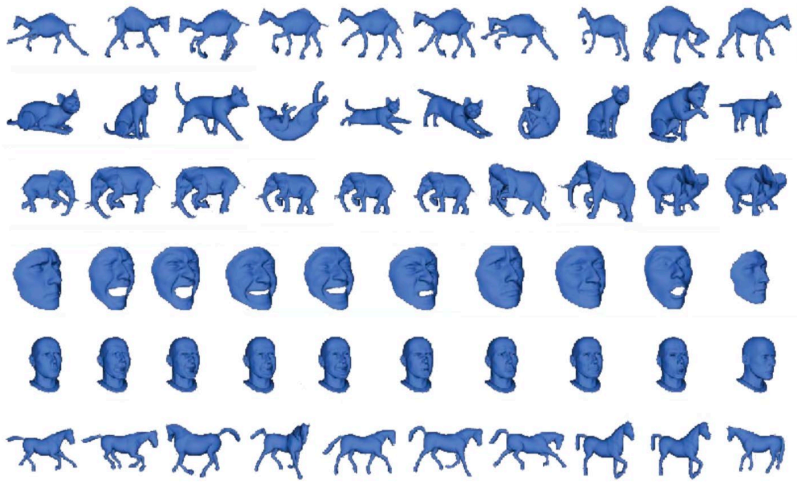
Figure from Edelsbrunner, Harer, *Computational Topology*

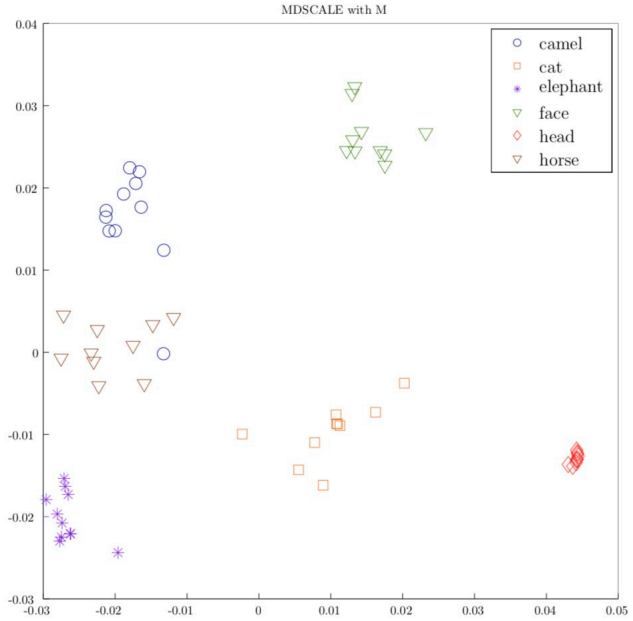
- Theorem (Cohen-Steiner, Edelsbrunner, Harer): $\mathcal{M}$ a triangulable space, $f, g: \mathcal{M} \to \mathbb{R}$ "nice". Then,
$$d_B\big(D_k(f), D_k(g)\big) \leq \|f - g\|_\infty.$$

Here D_k is the persistence diagram in dimension k, $\|\cdot\|_\infty$ is the max-norm, and d_B is the bottleneck distance (TBD)

Stability of persistent homology

- We also have stability when building filtered complexes from metric data

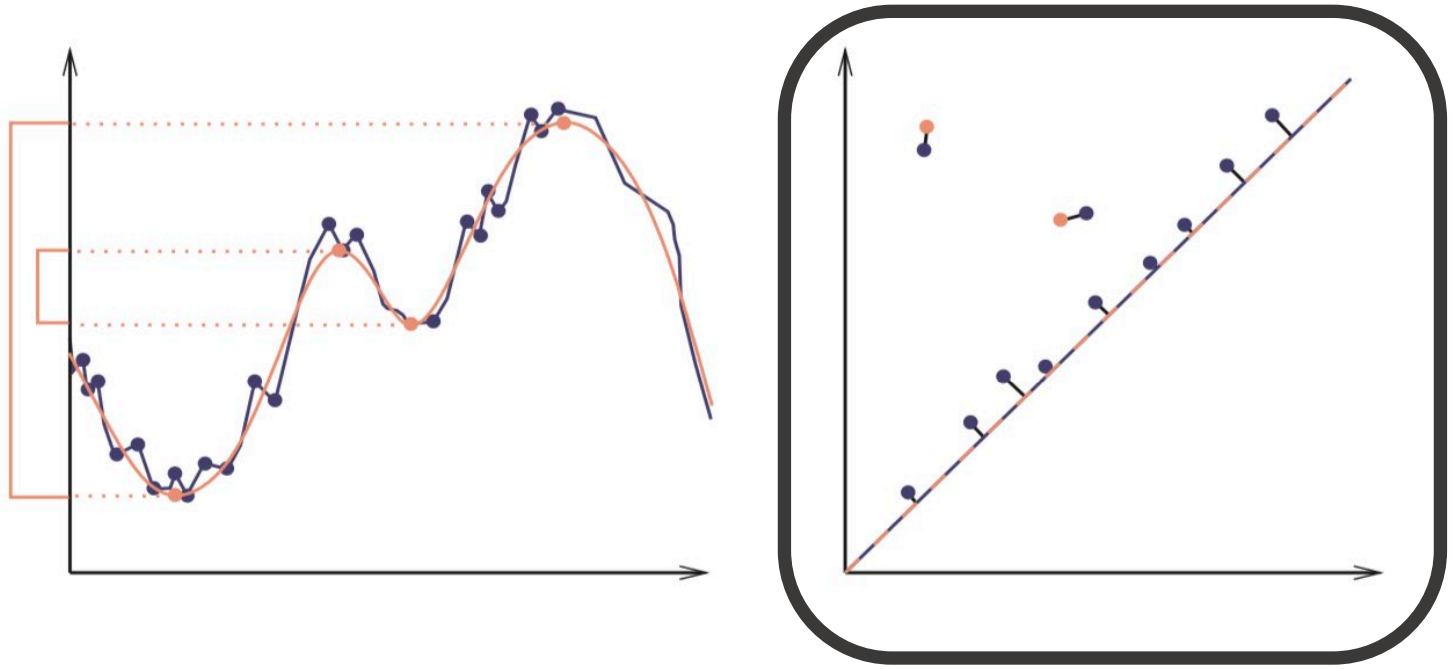




- Theorem (Chazal, Cohen-Steiner, Guibas, Mémoli, Oudot): $(X, d_X), (Y, d_Y)$ finite metric spaces. Then,
$$d_B\big(D_k(VR(X)), D_k(VR(Y))\big) \leq 2d_{GH}(X, Y).$$

Here d_{GH} is the Gromov-Hausdorff distance between metric spaces (TBD)

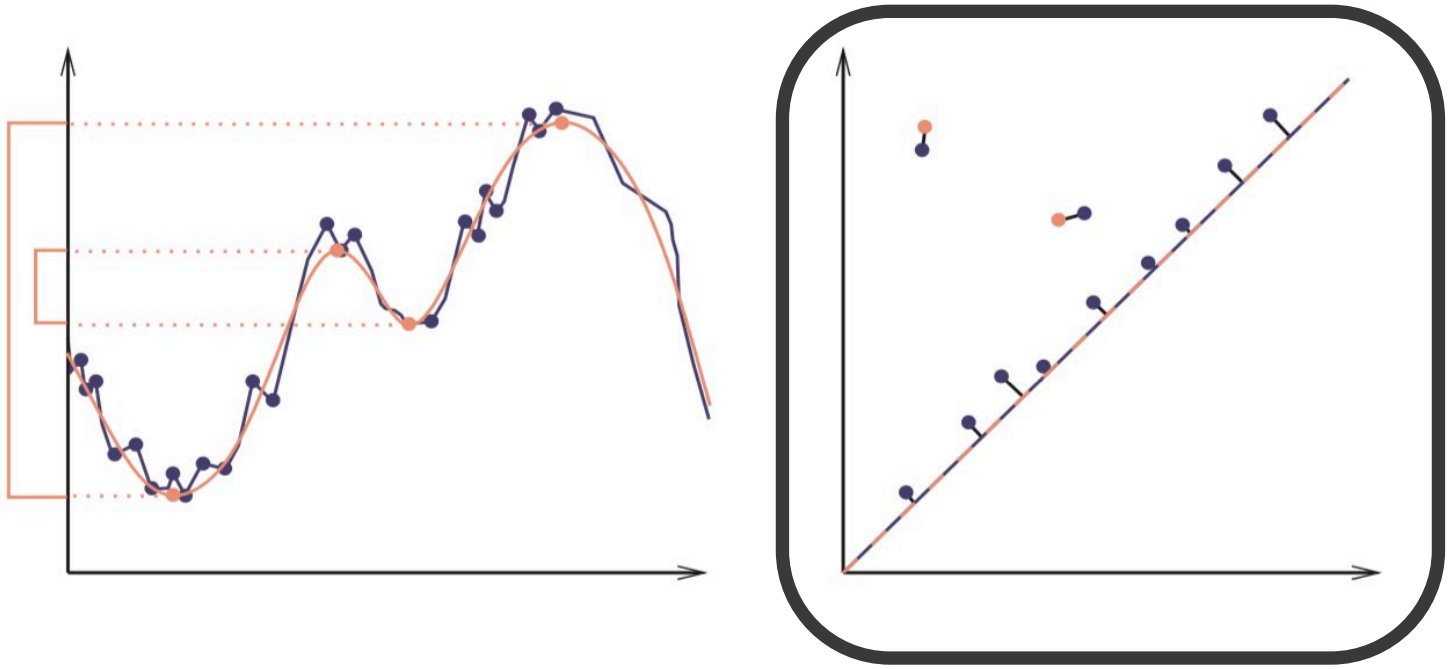
The bottleneck distance



Method for comparing persistence diagrams:

- Choose a bijection, matching points to the diagonal if needed
- Compute the l^{∞} distance between each pair of matched points (max-norm)
- Take worst case matching cost
- Optimize over all possible bijections

The bottleneck distance



A, B two persistence diagrams.

$$d_B(A,B) := \inf\{ \sup_{x \in A \cup \Delta} \|x - \phi(x)\|_\infty : \phi : A \cup \Delta \to B \cup \Delta \text{ a bijection}\}$$

Gromov-Hausdorff stability

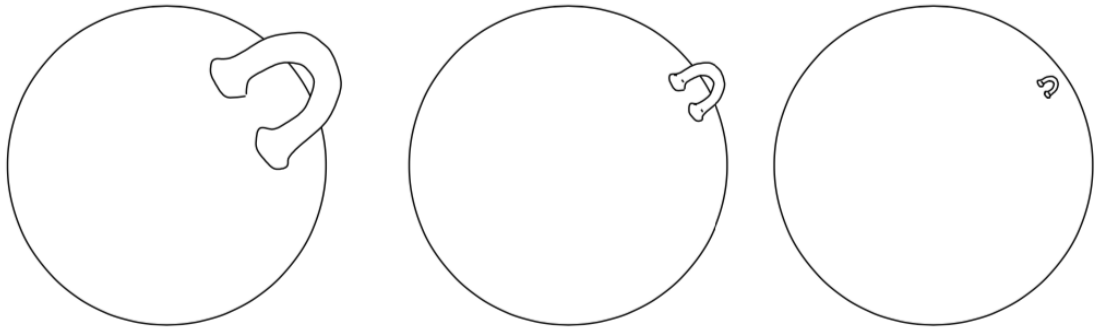


Figure from Burago, Burago, Ivanov, *A course in metric geometry*

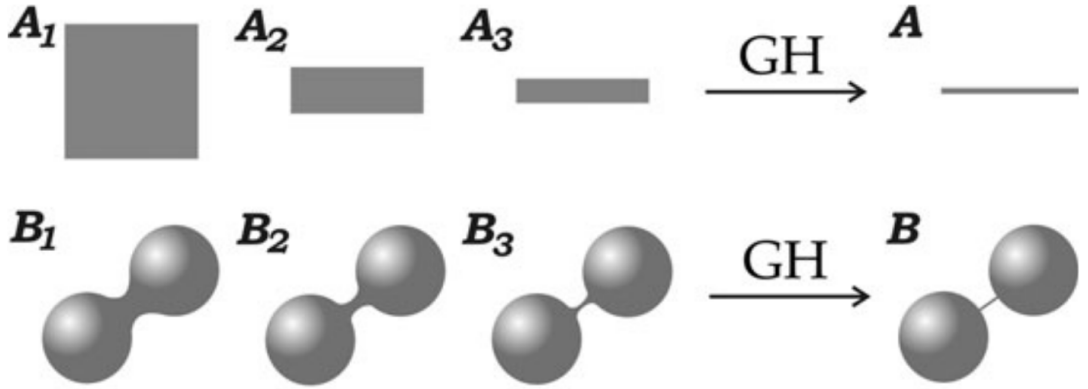


Figure from Sormani, *How Riemannian manifolds converge*

The space of metric spaces is a metric space under the Gromov-Hausdorff distance

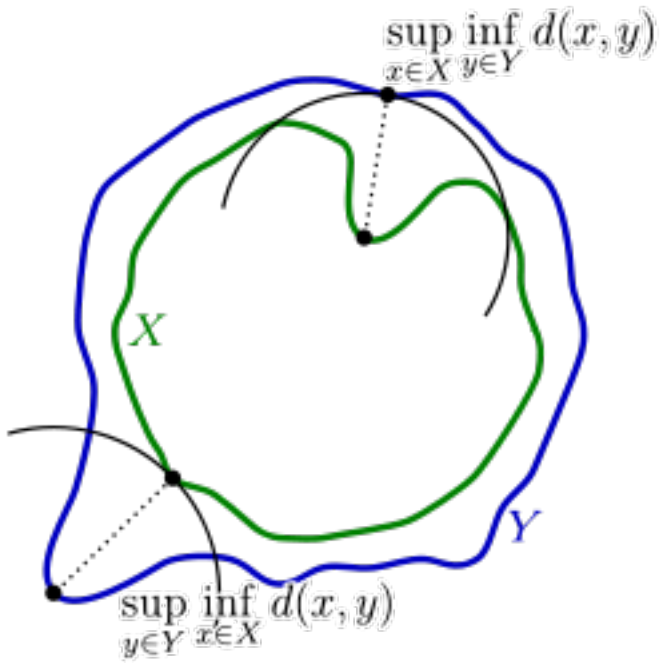
- Used to study convergence of spaces
- Popularized by Gromov in the context of geometric group theory

Gromov-Hausdorff stability

X, Y subsets of a finite metric space (Z, d)

Hausdorff distance:

$$d_H(X,Y) = \max(\max_{x\in X} \min_{y\in Y} d(x,y),\ \max_{y\in Y} \min_{x\in X} d(x,y))$$

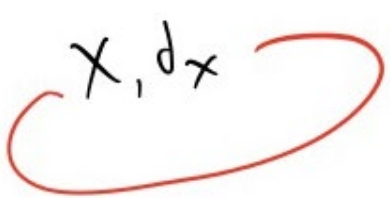


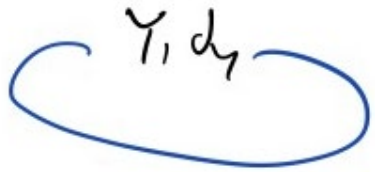
$(X, d_X), (Y, d_Y)$ finite metric spaces

Gromov-Hausdorff distance:

$d_{GH}(X,Y) = \inf_{Z,\phi_X,\phi_Y} d_H^Z(\phi_X(X), \phi_Y(Y))$, where

ϕ_X, ϕ_Y are isometric embeddings



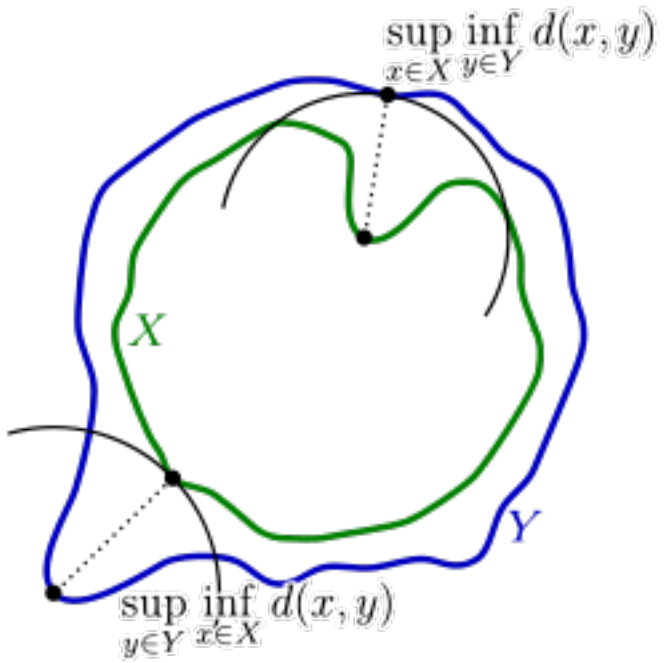


Gromov-Hausdorff stability

X, Y subsets of a finite metric space (Z, d)

Hausdorff distance:

$$d_H(X,Y) = \max(\max_{x\in X} \min_{y\in Y} d(x,y),\ \max_{y\in Y} \min_{x\in X} d(x,y))$$

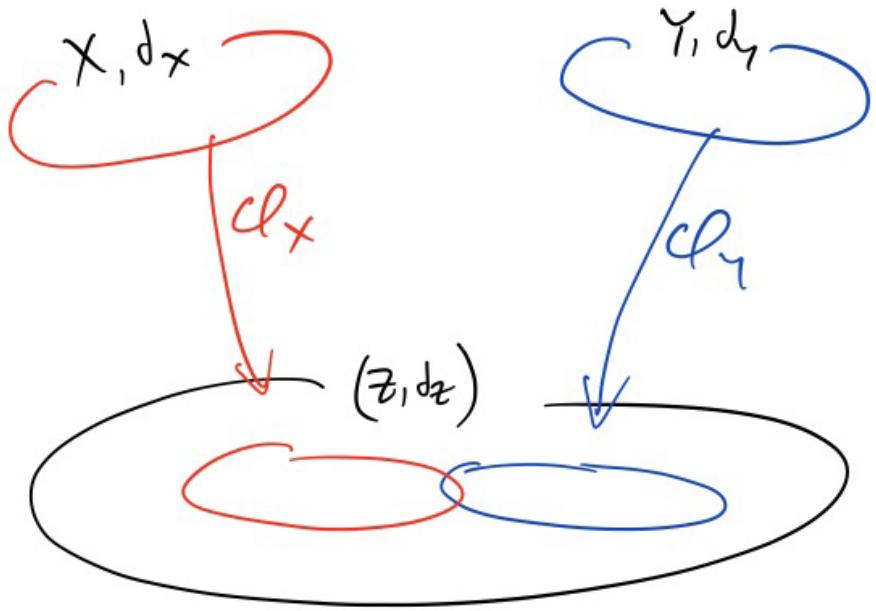


$(X, d_X), (Y, d_Y)$ finite metric spaces

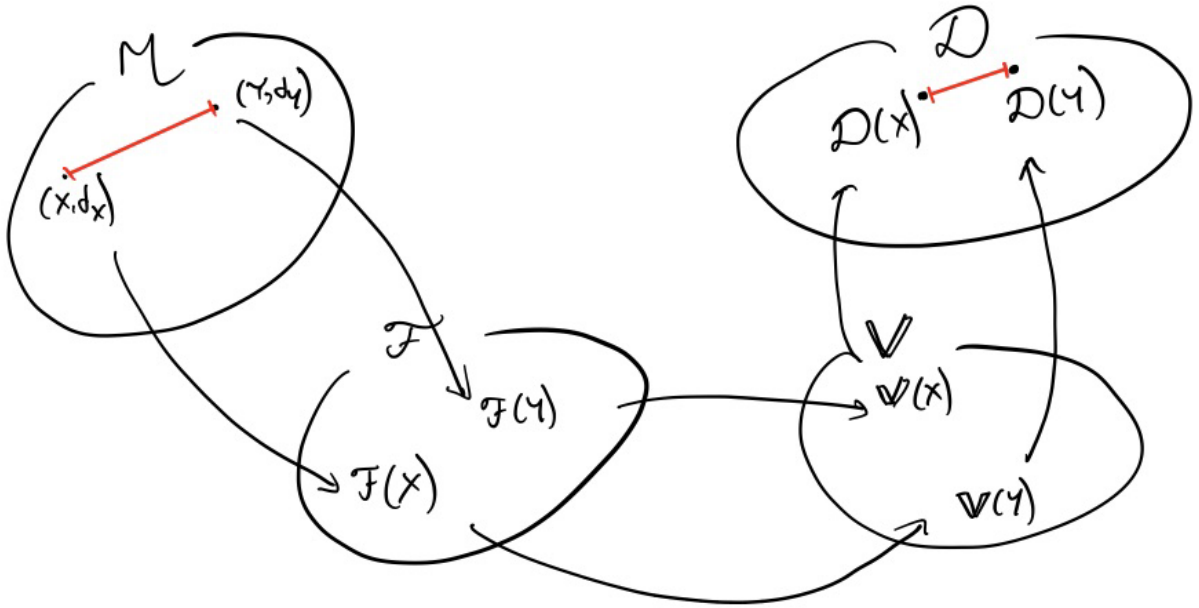
Gromov-Hausdorff distance:

$d_{GH}(X,Y) = \inf_{Z,\phi_X,\phi_Y} d_H^Z(\phi_X(X), \phi_Y(Y))$, where

ϕ_X, ϕ_Y are isometric embeddings



Gromov-Hausdorff stability



$\mathcal{M}$ = finite metric spaces
$\mathcal{F}$ = filtered simplicial complexes
$\mathbb{V}$ = persistent vector spaces
$\mathbb{D}$ = persistence diagrams

- Theorem (Chazal, Cohen-Steiner, Guibas, Mémoli, Oudot): $(X, d_X), (Y, d_Y)$ finite metric spaces. Then,

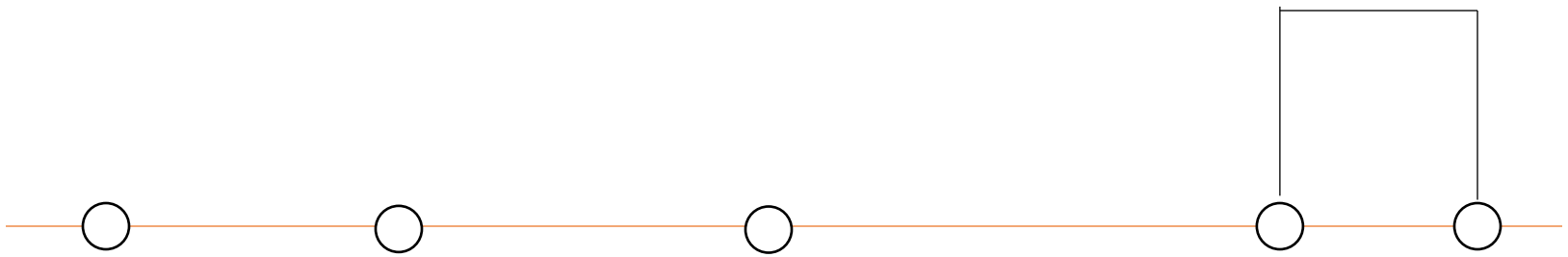
$$d_B\big(D_k(VR(X)), D_k(VR(Y))\big) \leq 2d_{GH}(X, Y).$$

$O(n^3)$ with Hungarian algorithm, $O\left(n^{3/2}\right)$ state-of-the-art (Kerber, Morozov, Nigmetov 2017)

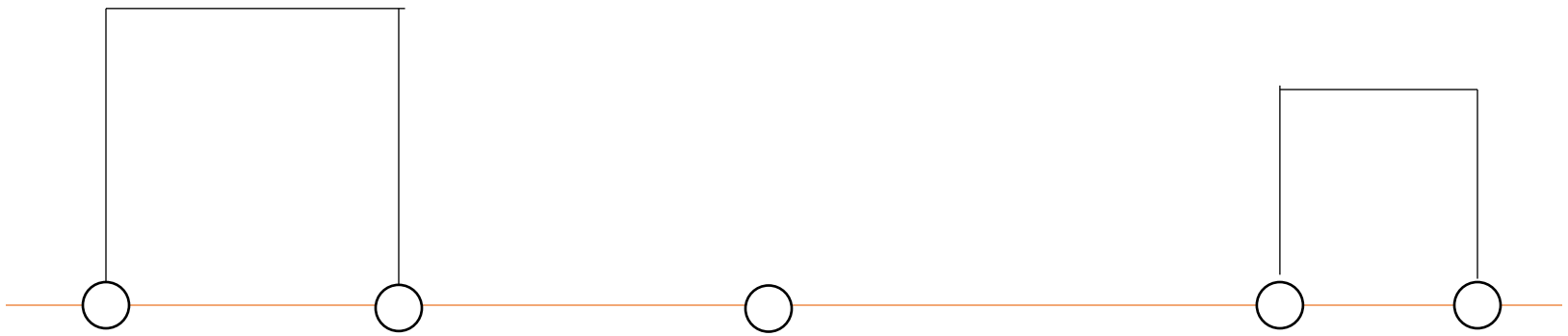
NP-hard

0-dimensional persistence and single linkage clustering

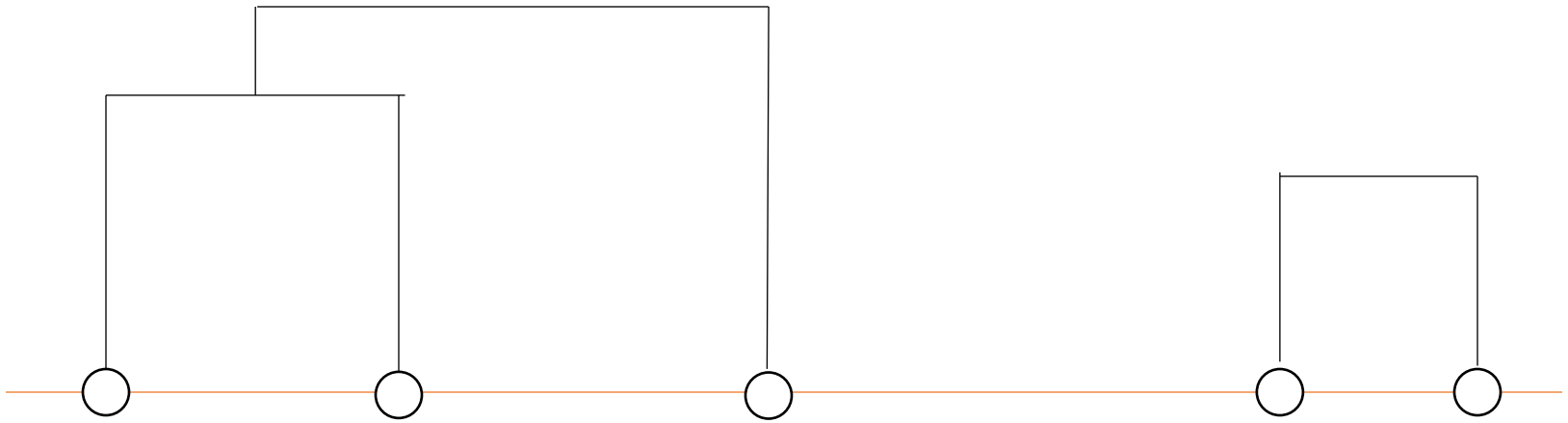
0-dimensional persistence and single linkage clustering



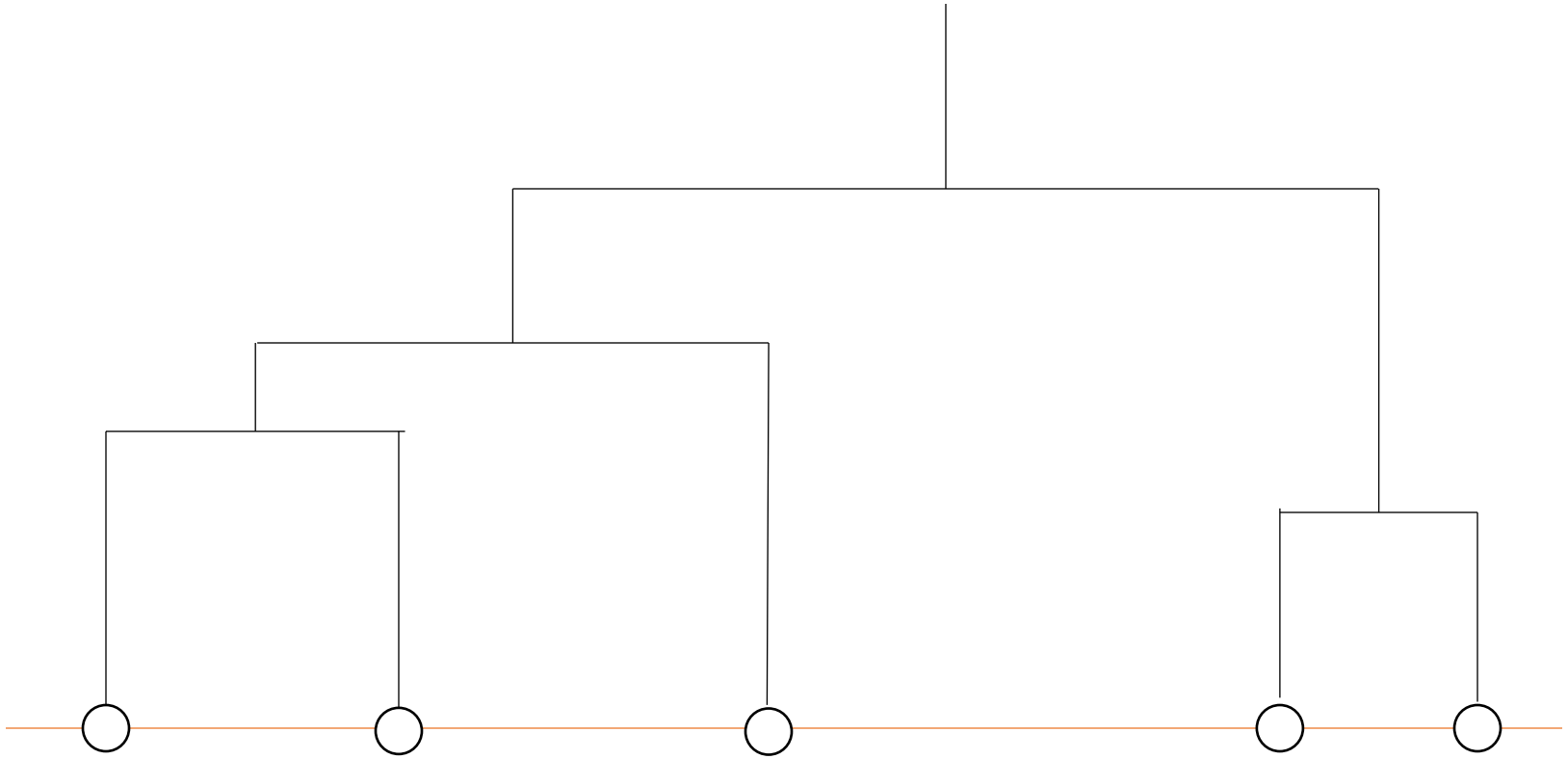
0-dimensional persistence and single linkage clustering



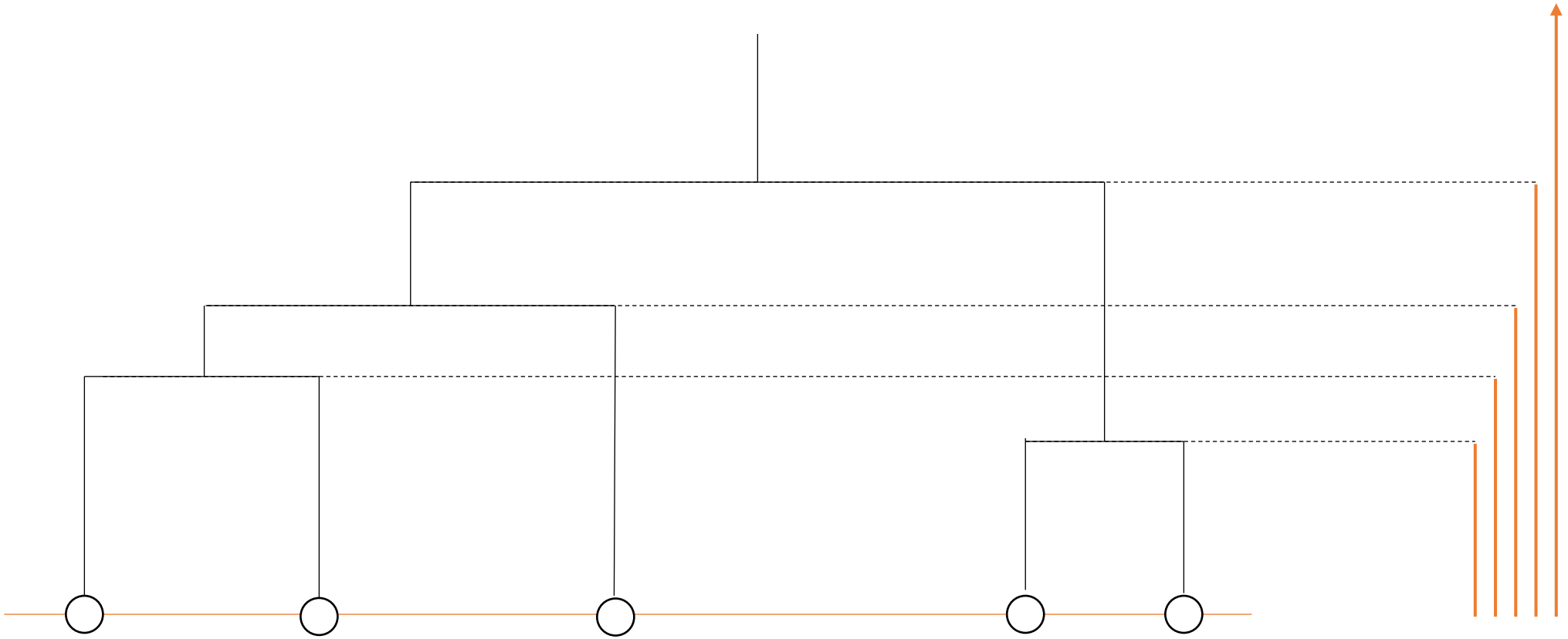
0-dimensional persistence and single linkage clustering



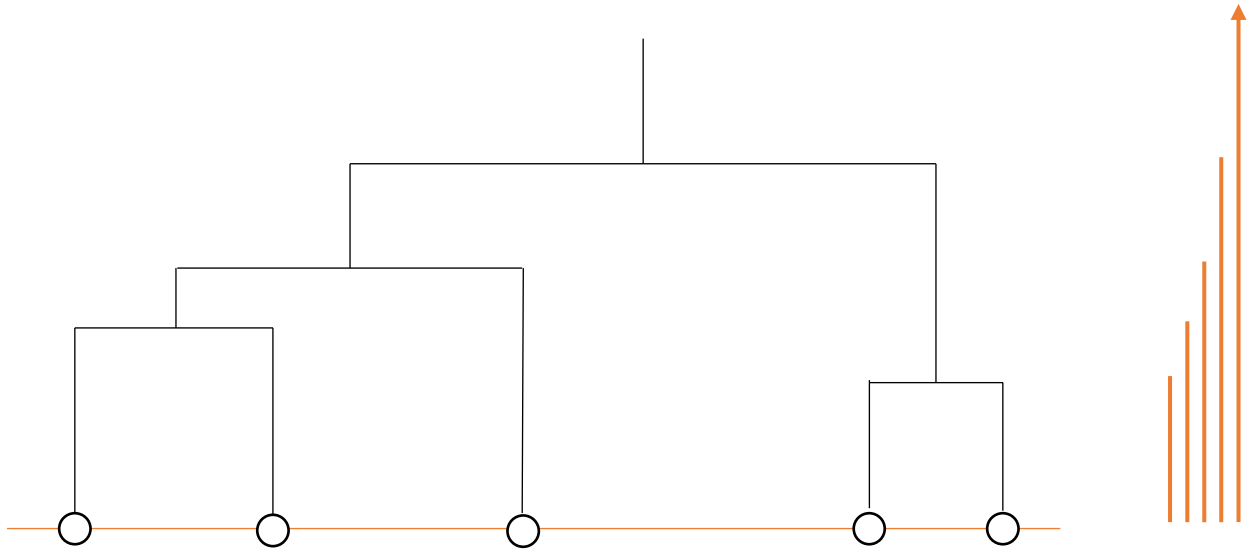
0-dimensional persistence and single linkage clustering



0-dimensional persistence and single linkage clustering



0-dimensional persistence and single linkage clustering



- Fact: { dendrograms } ↔ {ultrametric spaces}
- Theorem (Carlsson, Mémoli): Let (X, u_X) denote the output of SLHC on (X, d_X). Then,

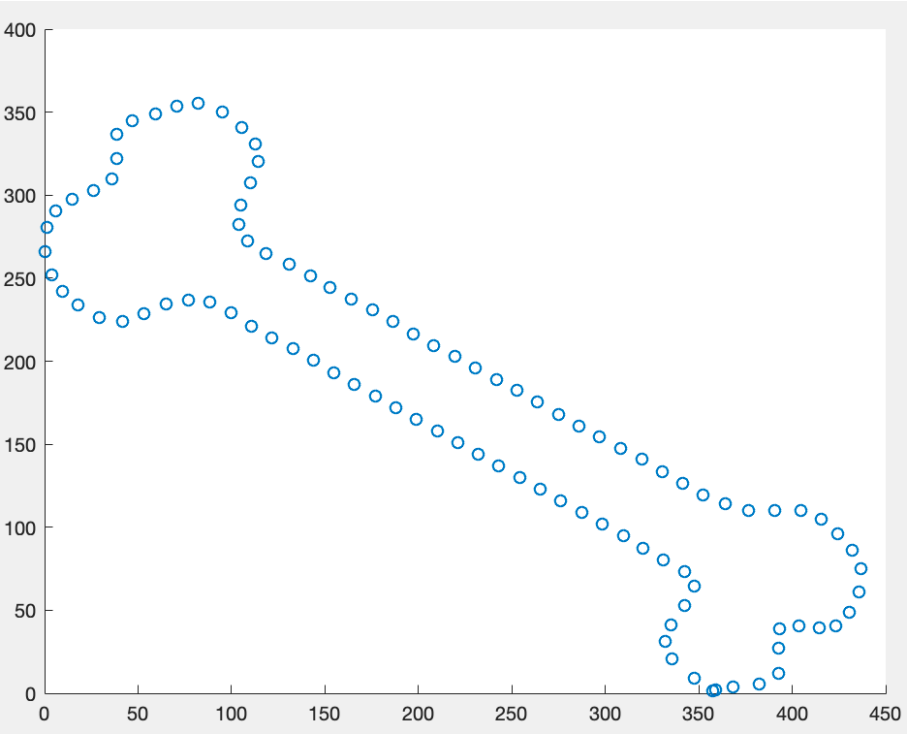
$$d_{GH}\big((X, u_X), (Y, u_Y)\big) \leq d_{GH}\big((X, d_X), (Y, d_Y)\big).$$

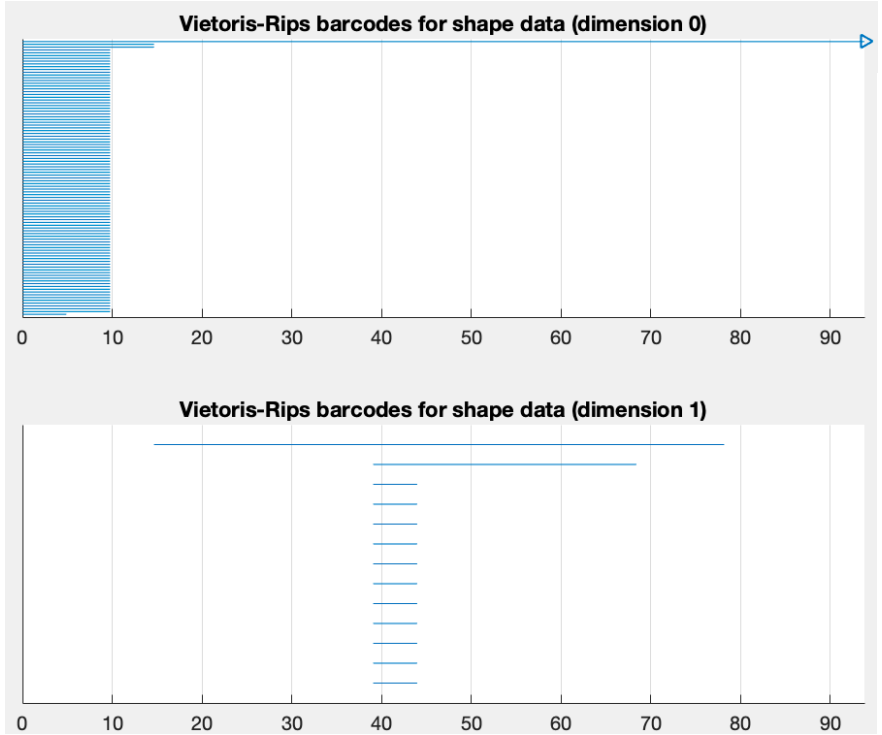
- (Tradeoff) 0-dimensional persistence loses information but permits comparison via d_B (easier than d_{GH})

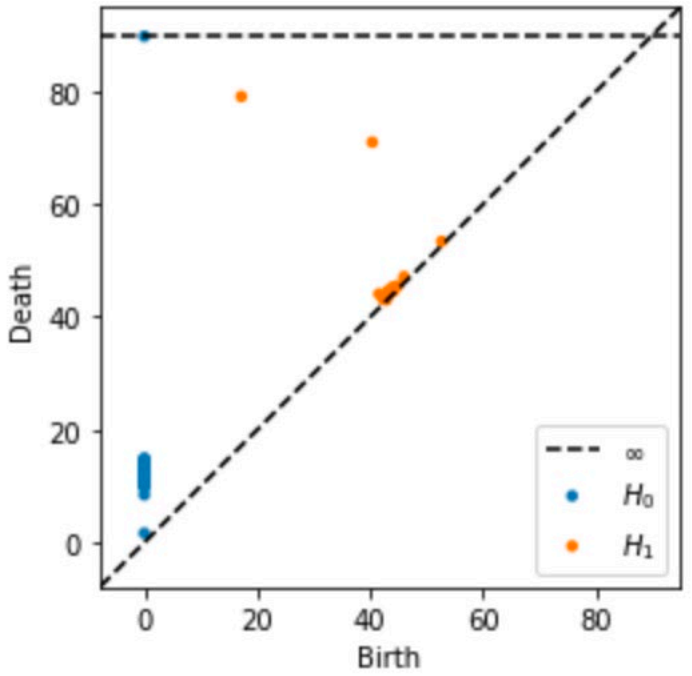
Software packages

- Javaplex - http://appliedtopology.github.io/javaplex/
- Ripser - https://github.com/Ripser/ripser
  - Ripser.py - https://github.com/scikit-tda/ripser.py
  - Ripser live - https://live.ripser.org/
- Ripser++ - https://github.com/simonzhang00/ripser-plusplus
- Dionysus 2 - https://mrzv.org/software/dionysus2/
- Gudhi - https://gudhi.inria.fr/python/latest/
- DIPHA - https://github.com/DIPHA/dipha
- PHAT - https://bitbucket.org/phat-code/phat
- Eirene - https://github.com/Eetion/Eirene.jl
- …many others

Demo – Javaplex and Ripser







The End