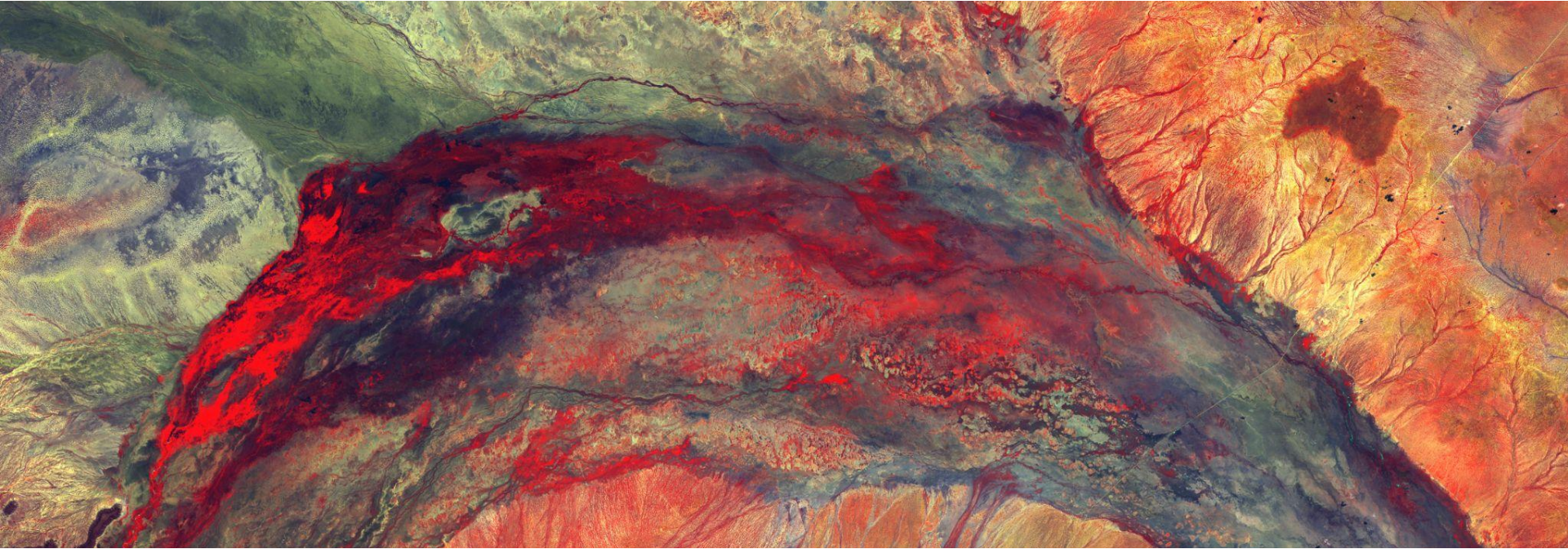


# EDS 223: Geospatial Analysis & Remote Sensing

## Week 3



# Welcome!

- **Week 2 recap**
- **Building a spatial analysis workflow**
  - Subsetting
  - Aggregating
  - Summarizing
  - Simplifying

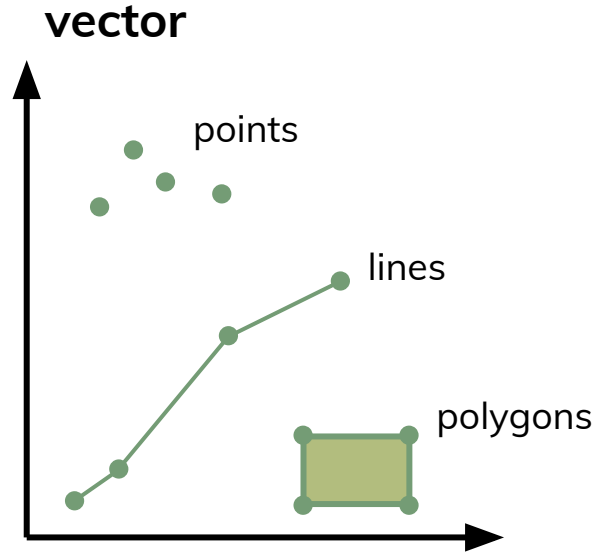
# How to get unstuck

Start here

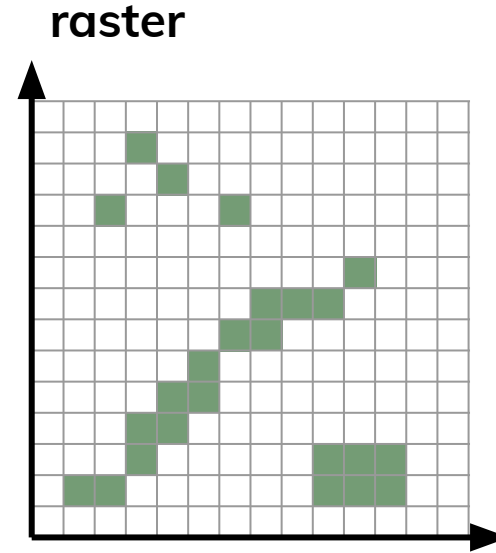


Resource	Steps
Yourself	<ul style="list-style-type: none"><li>• Review the lecture/lab/discussion materials</li><li>• Review the background reading</li><li>• Google!</li></ul>
Your peers	<ul style="list-style-type: none"><li>• Talk to a friend</li><li>• Ask the #eds-223-geospatial Slack channel</li></ul>
TA	<ul style="list-style-type: none"><li>• Ask questions in discussion section</li><li>• Attend office hours</li><li>• Send a message over Slack</li></ul>
Instructor	<ul style="list-style-type: none"><li>• Attend office hours</li><li>• Send a message over Slack</li></ul>

# Spatial data models



•• discrete

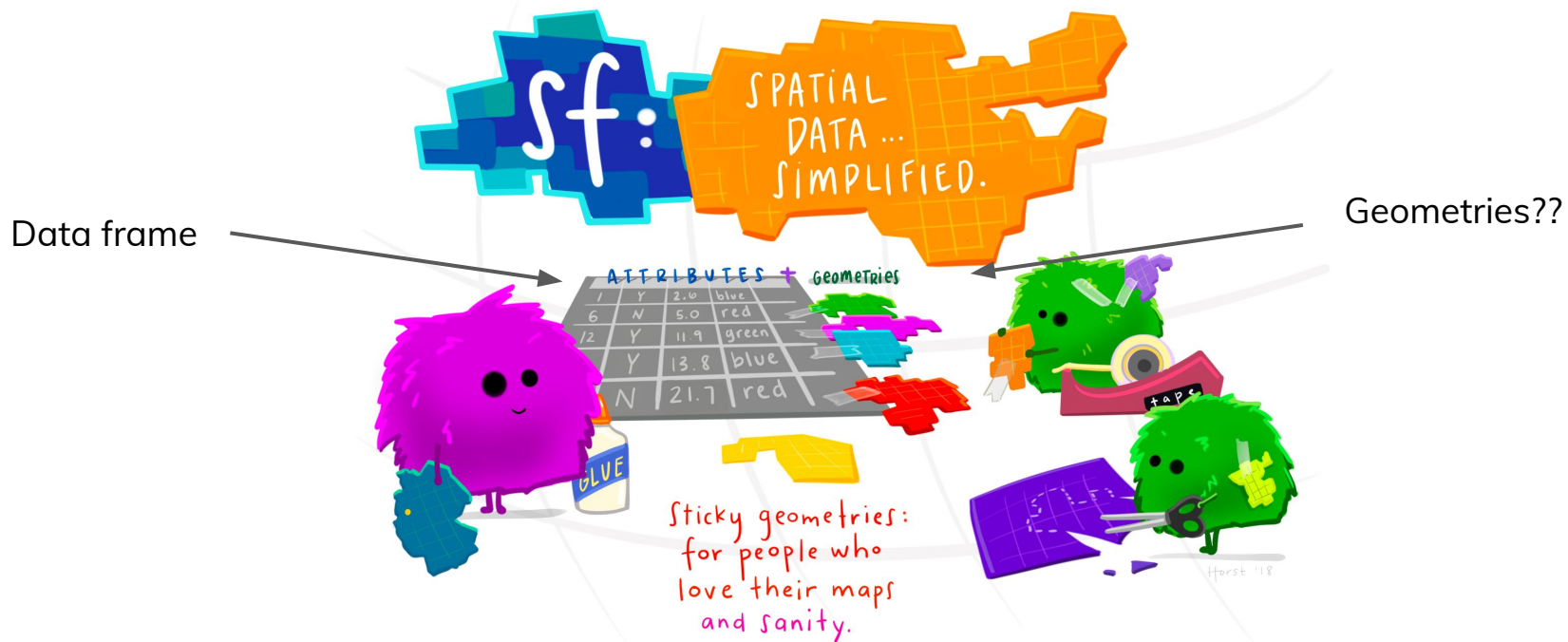


continuous

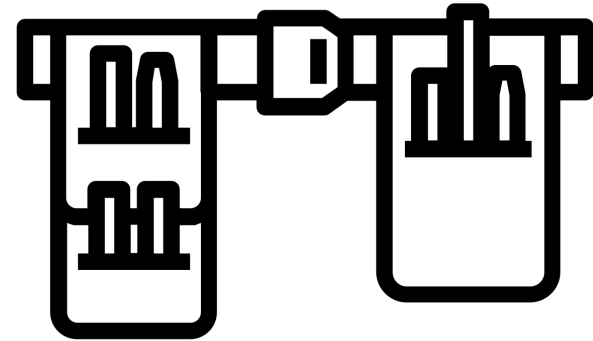


geometries

# Simple features: **sf**

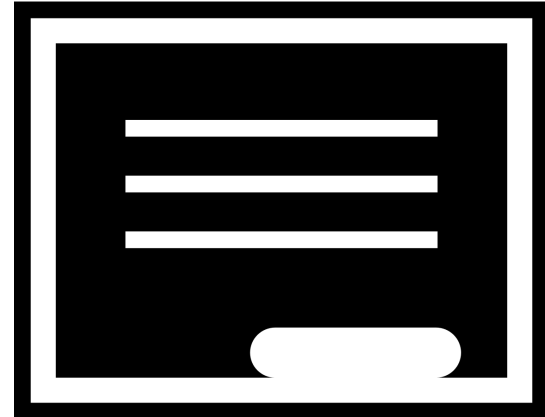


# Toolbelt for solving spatial problems



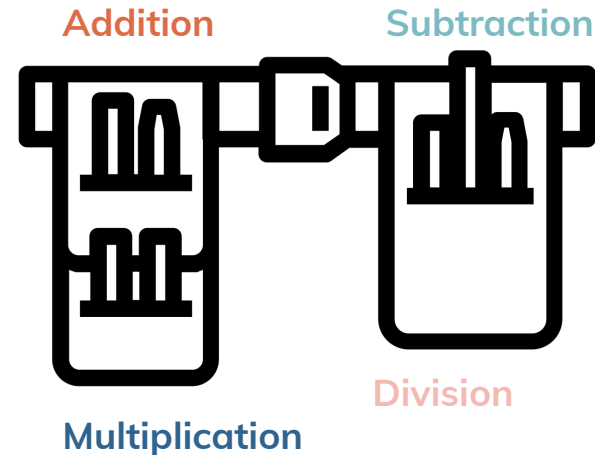
# Toolbelt for solving spatial problems

There is a group of 10 people who are ordering pizza. If each person gets 2 slices and each pizza has 4 slices, how many pizzas should they order?



# Toolbelt for solving spatial problems

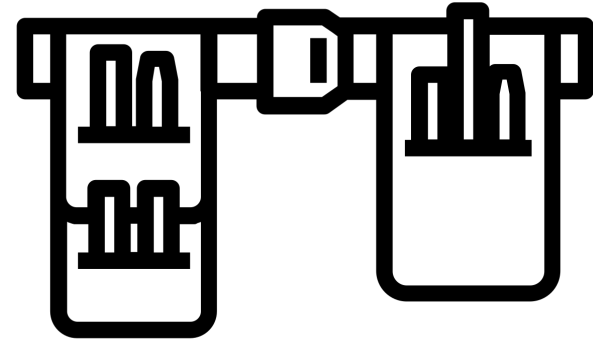
There is a group of 10 people who are ordering pizza. If each person gets 2 slices and each pizza has 4 slices, how many pizzas should they order?





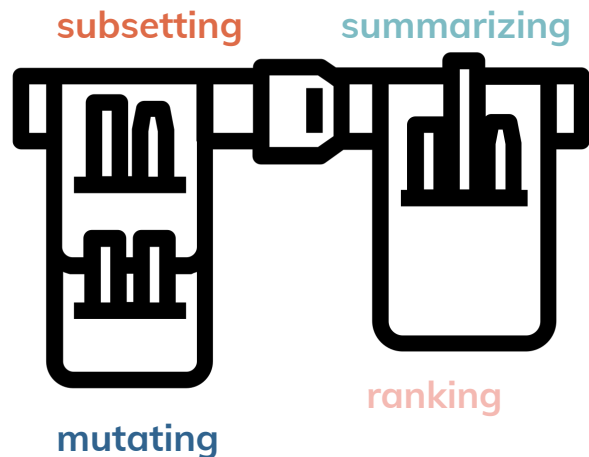
# Toolbelt for solving spatial problems

What is the life expectancy of the country in Asia with the highest population density?



# Toolbelt for solving spatial problems

What is the life expectancy of the country in Asia with the highest population density?



# New tools for a new data type

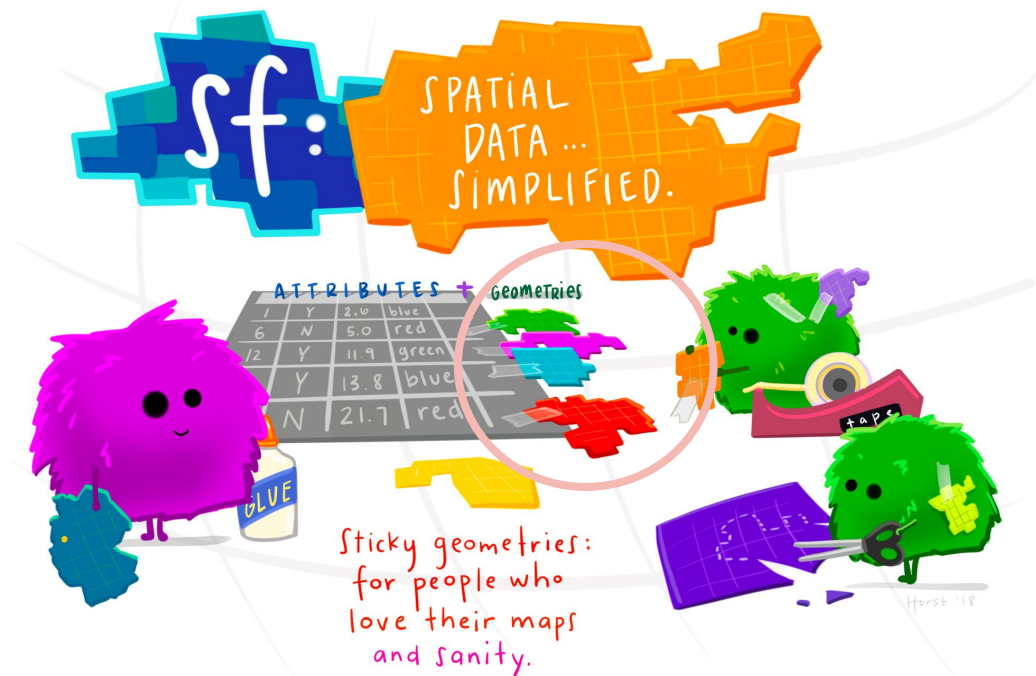
dplyr : go wrangling



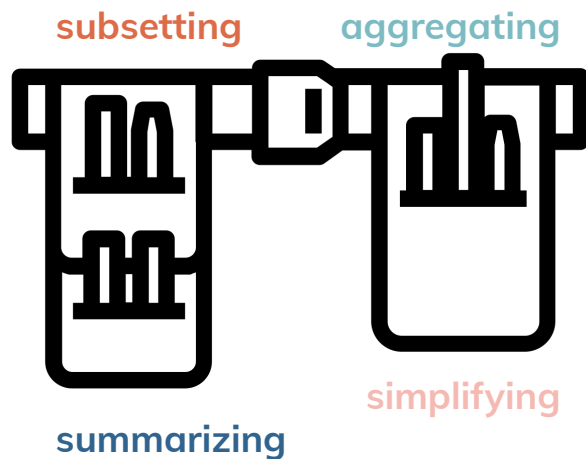
# New tools for a new data type



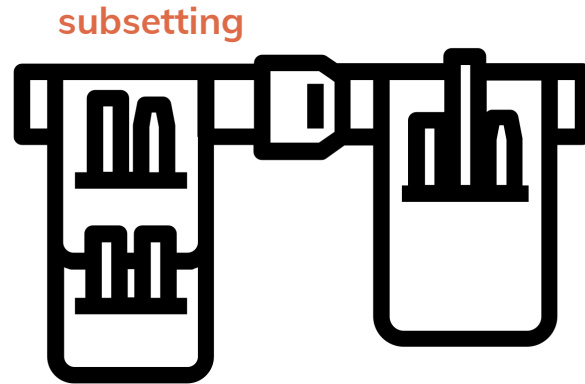
# New tools for a new data type



# Toolbelt for solving spatial problems



# Toolbelt for solving spatial problems



# New tools for a new data type

dplyr: go wrangling



**dplyr::filter()** KEEP ROWS THAT satisfy your **CONDITIONS**

```
filter(df, type == "otter" & site == "bay")
```

Annotations: "keep rows from..." points to 'df', "this data..." points to 'df', "ONLY IF..." points to 'type == "otter"', "AND" points to '&', "site is 'bay'" points to 'site == "bay"'.

type	food	site
otter	urchin	bay
Shark	seal	channel
otter	abalone	bay
otter	crab	wharf

Annotations: The second row (Shark, seal, channel) is crossed out with an orange line and an 'X'. The third row (otter, abalone, bay) has a purple checkmark. The fourth row (otter, crab, wharf) is crossed out with an orange line and an 'X'. The purple character has a checkmark next to the third row and an 'X' next to the fourth row.

@allison\_horst



# New tools for a new data type

How many mountains over 14K feet are in the United States?

# New tools for a new data type

How many mountains over 14K feet are in the United States?



Name	Country	Latitude, longitude
Machu Picchu	Peru	13.163°S, 72.545°W
...		...

# New tools for a new data type

How many mountains over 14K feet are in the United States?



Name	Latitude, longitude
Machu Picchu	13.163°S, 72.545°W
...	...

# New tools for a new data type

How many mountains over 14K feet are in the United States?



Name	Latitude, longitude
Machu Picchu	13.163°S, 72.545°W
...	...



# New tools for a new data type

How many mountains over 14K feet are in the United States?



Name	Latitude, longitude
Machu Picchu	13.163°S, 72.545°W
...	...



# New tools for a new data type

How many mountains over 14K feet are in the United States?

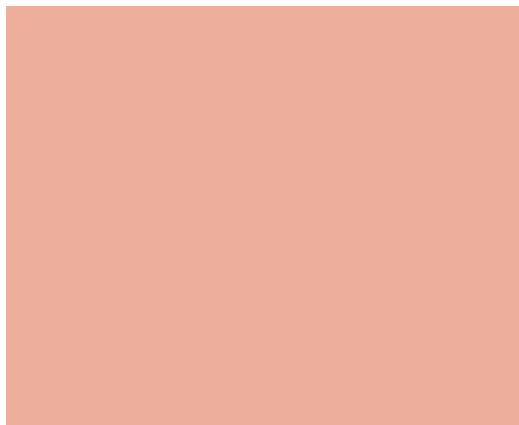


Name	Latitude, longitude
Machu Picchu	13.163°S, 72.545°W
...	...



**Geometry!**

# Topological relationships



# Topological relationships

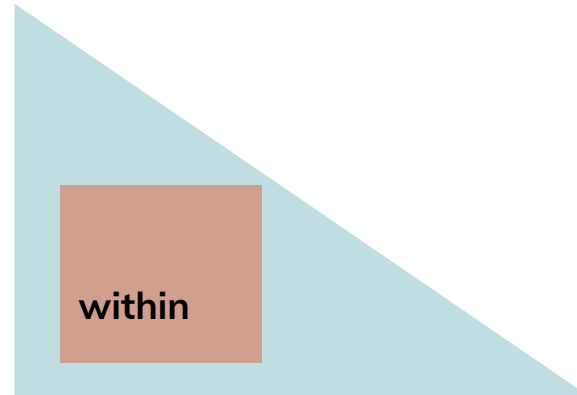




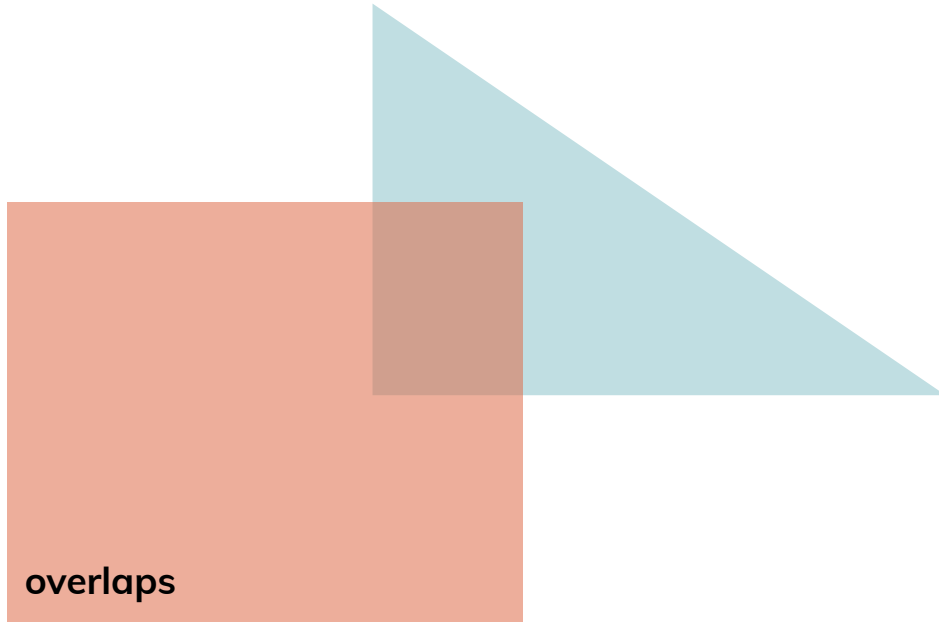
# Topological relationships



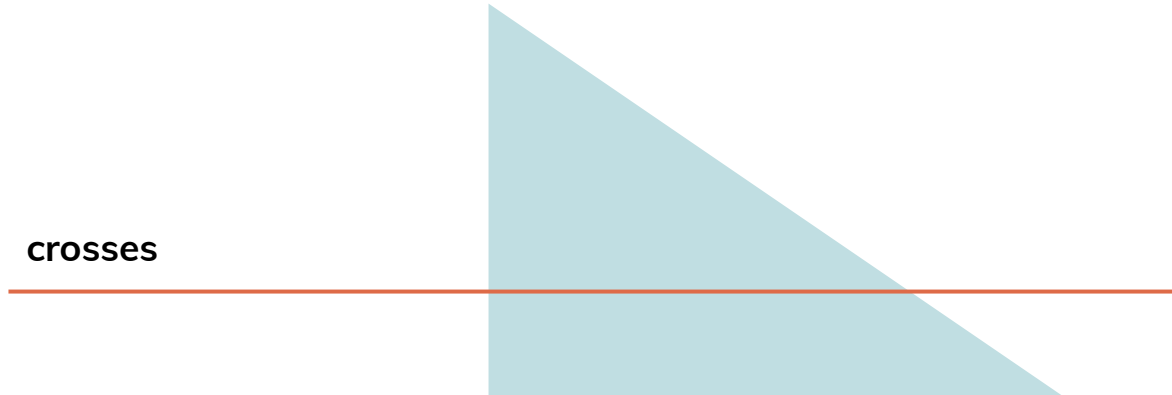
# Topological relationships



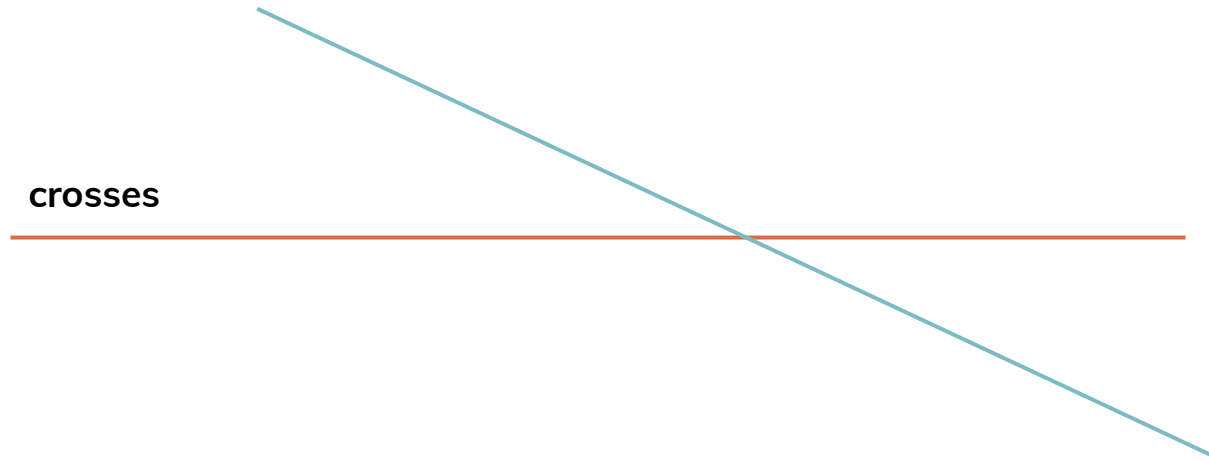
# Topological relationships



# Topological relationships

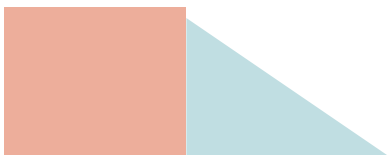


# Topological relationships

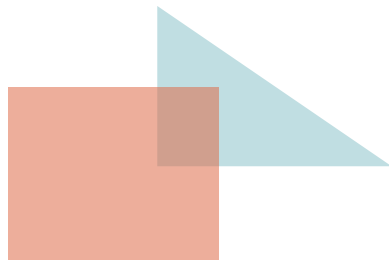


# Topological relationships

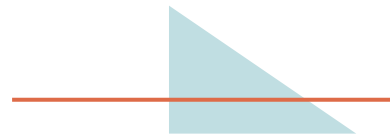
touches



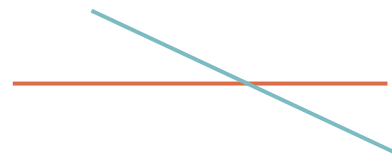
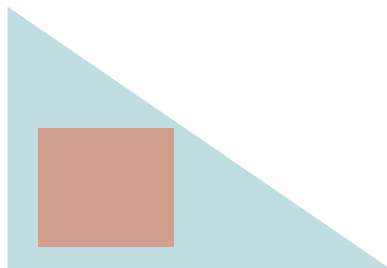
overlaps



crosses



within

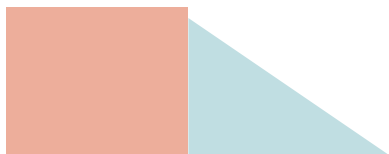


# Topological relationships

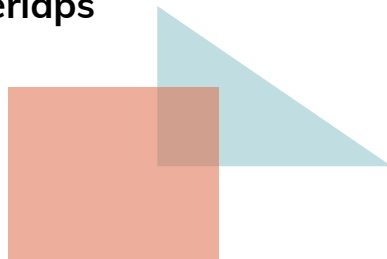
intersects

Yes or No

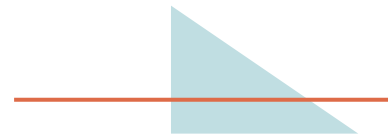
touches



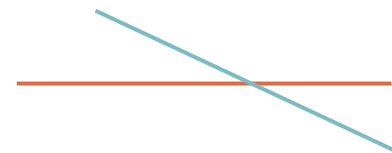
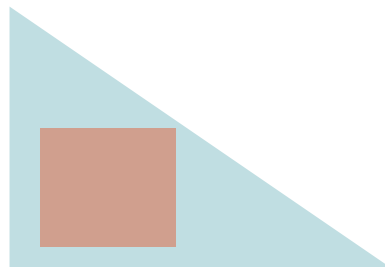
overlaps



crosses

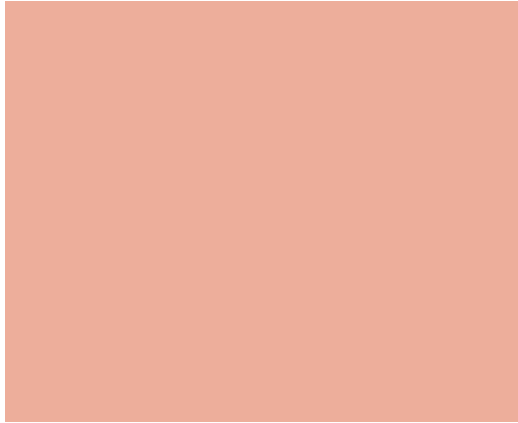


within



# Topological relationships

disjoint



Yes or No



# Topological relationships

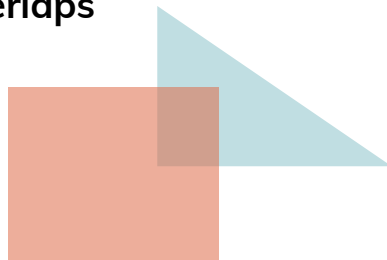
intersects

Yes or No

touches



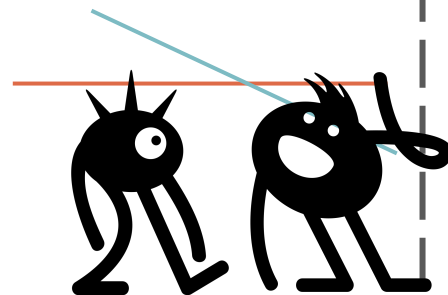
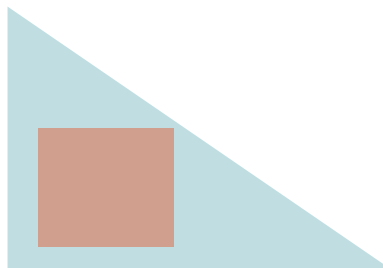
overlaps



crosses



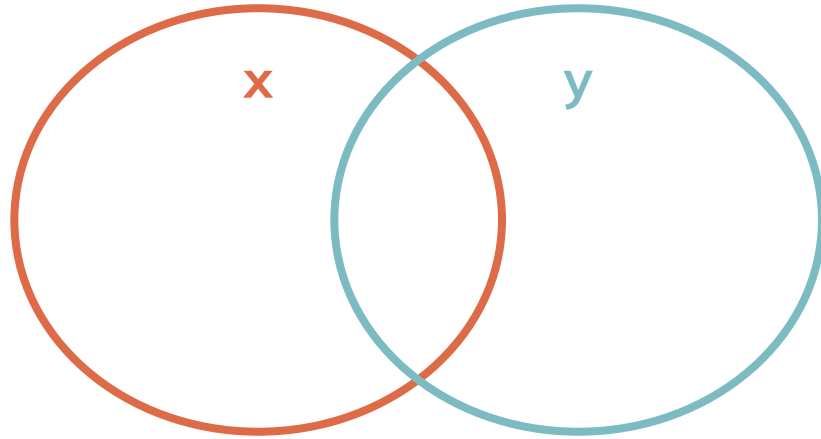
within



# Topological relationships

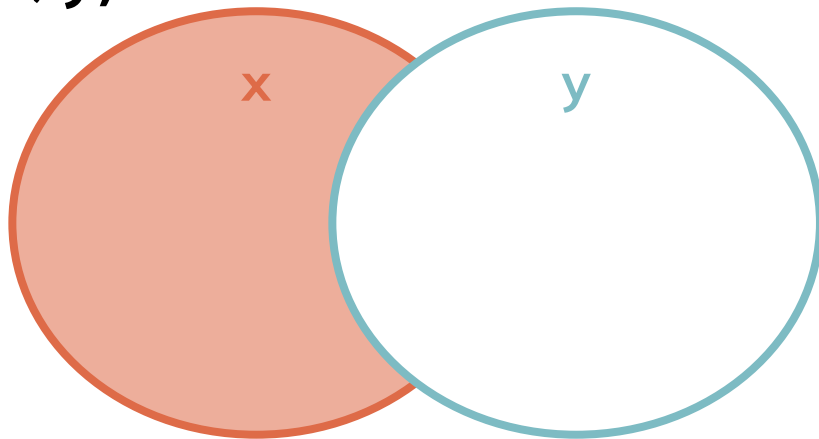


# Topological relationships: clipping



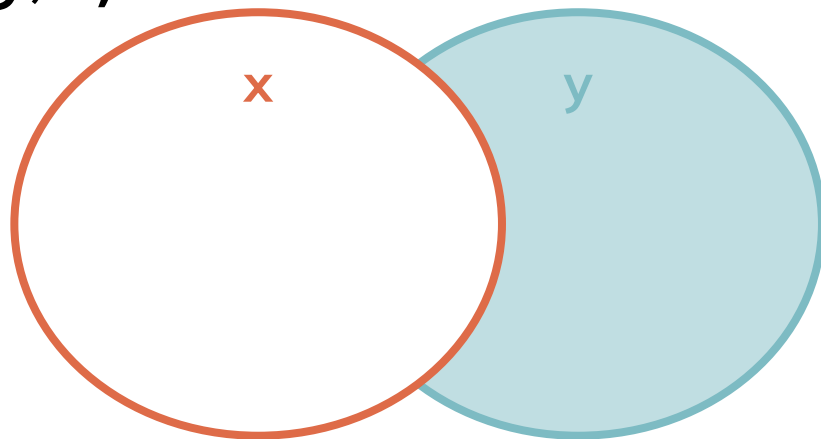
# Topological relationships: clipping

**difference (x, y)**



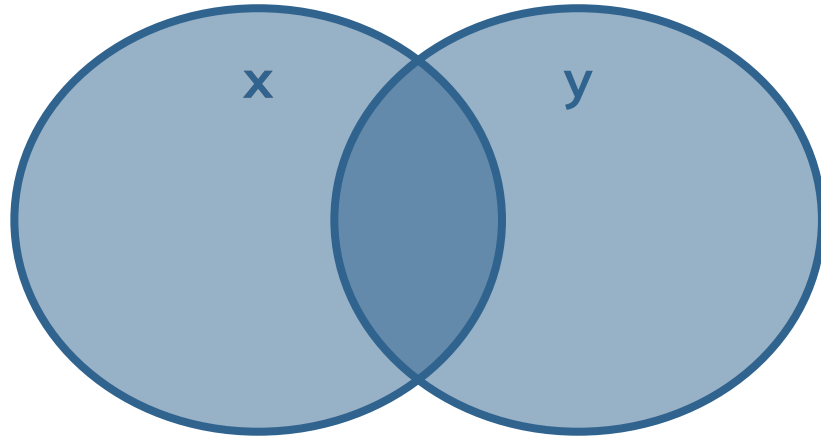
# Topological relationships: clipping

**difference (y, x)**



# Topological relationships: clipping

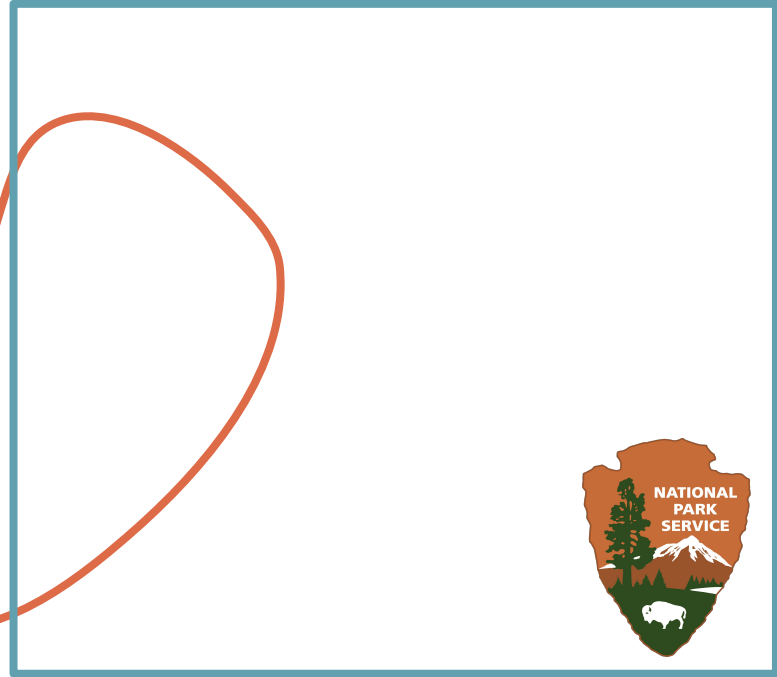
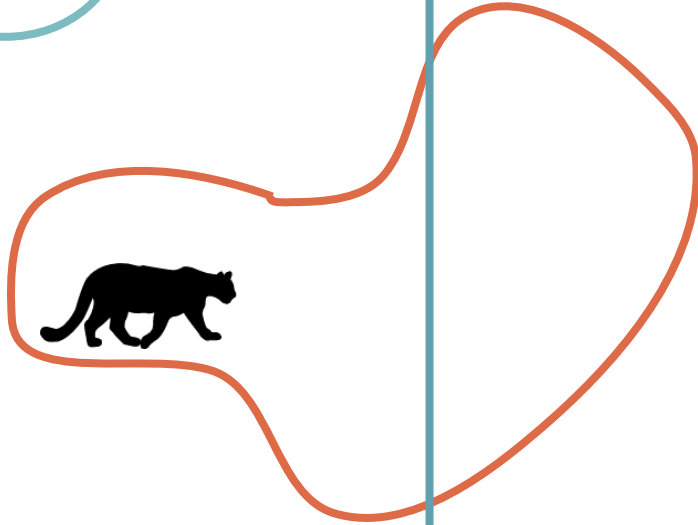
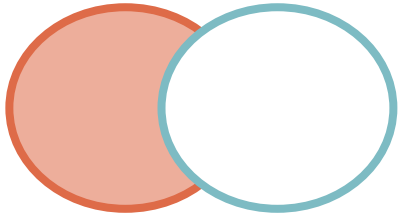
**union**



# Topological relationships

What proportion of a species' range is unprotected?

# Topological relationships



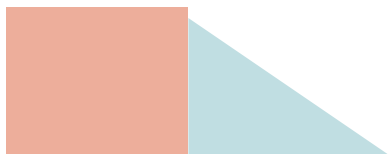


# Topological relationships

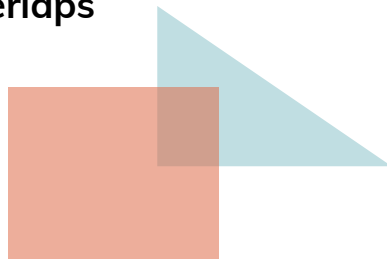
intersects

Yes or No

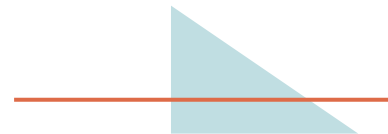
touches



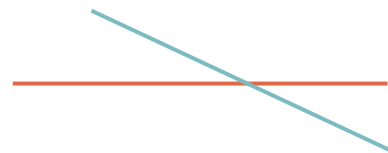
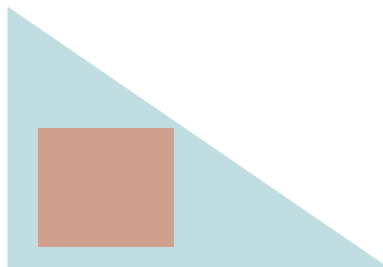
overlaps



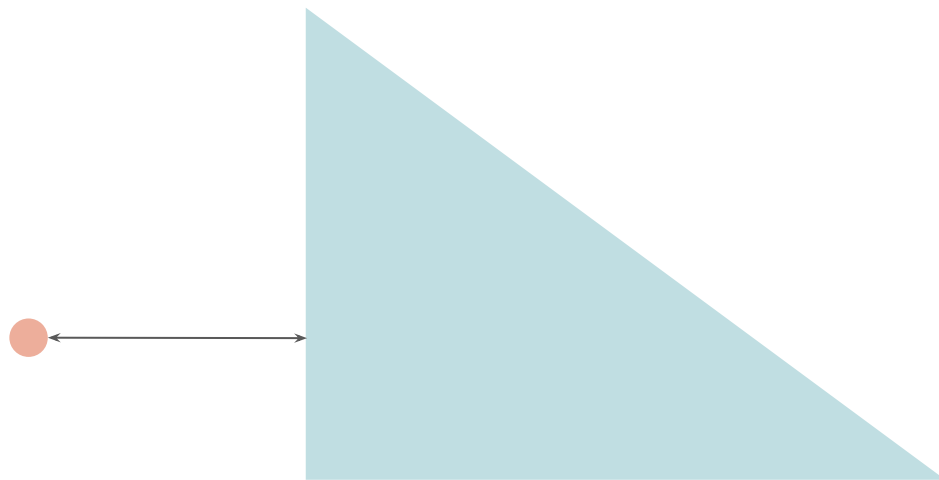
crosses



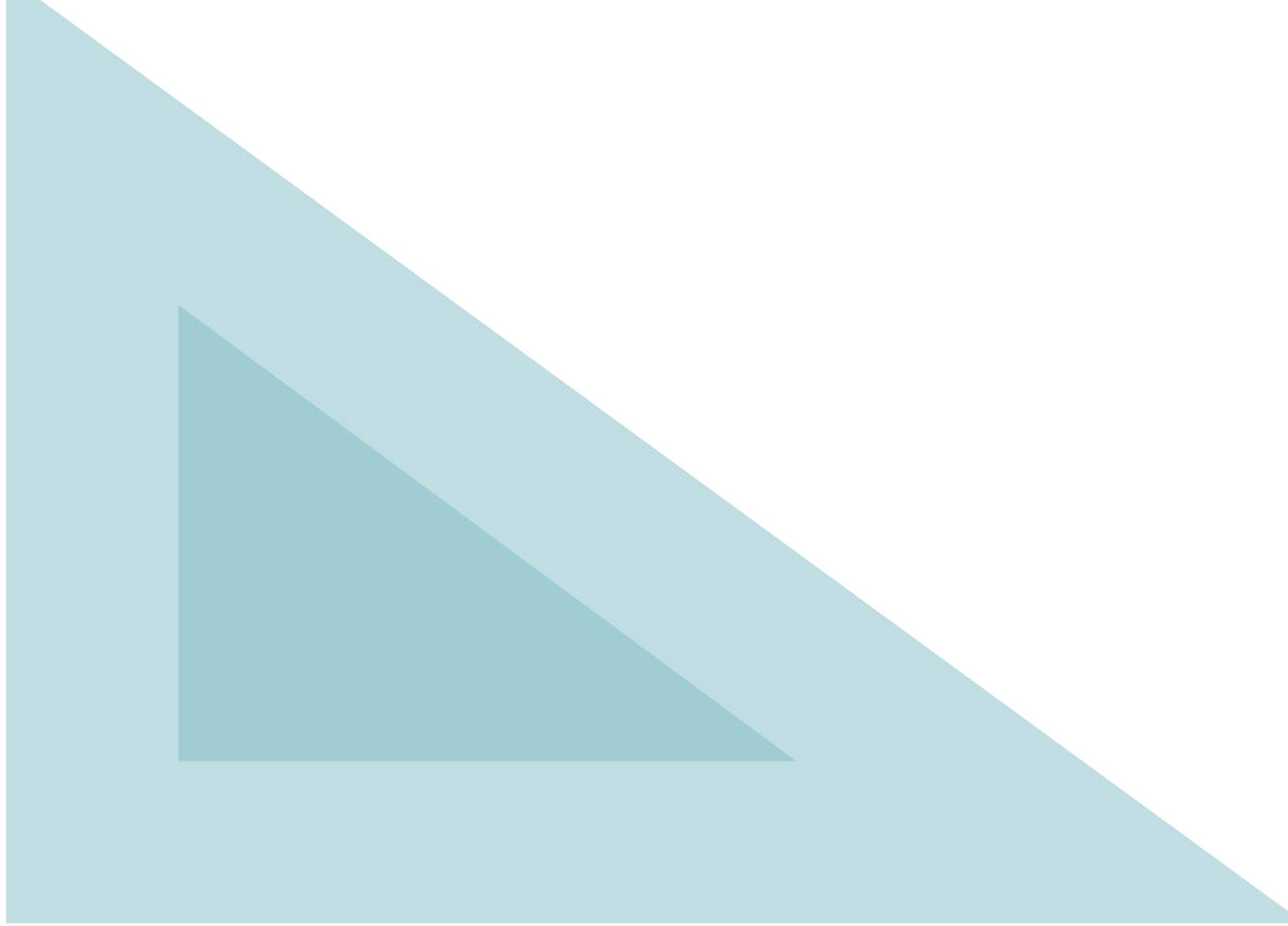
within



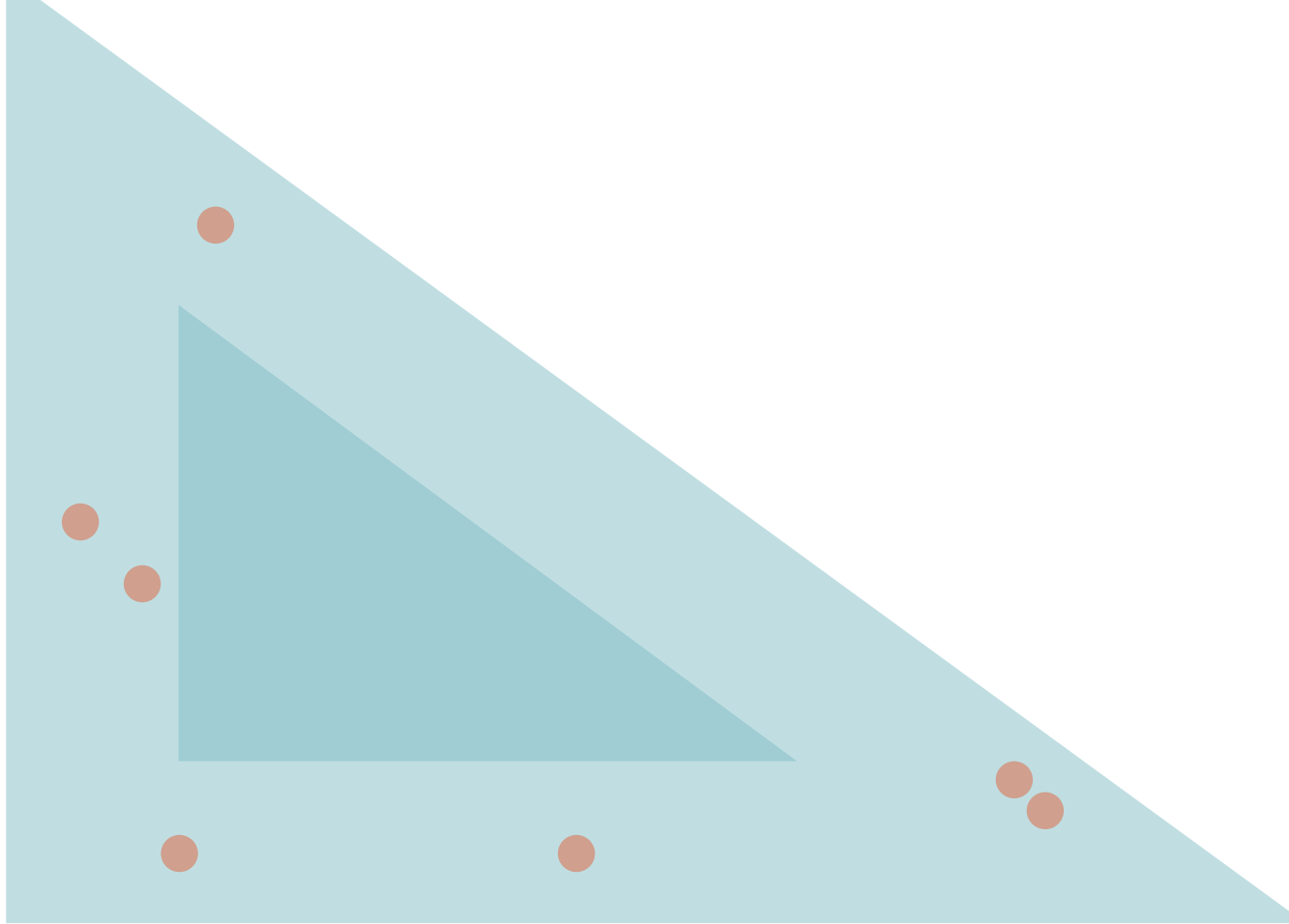
# Topological relationships



# Buffers



# Buffers

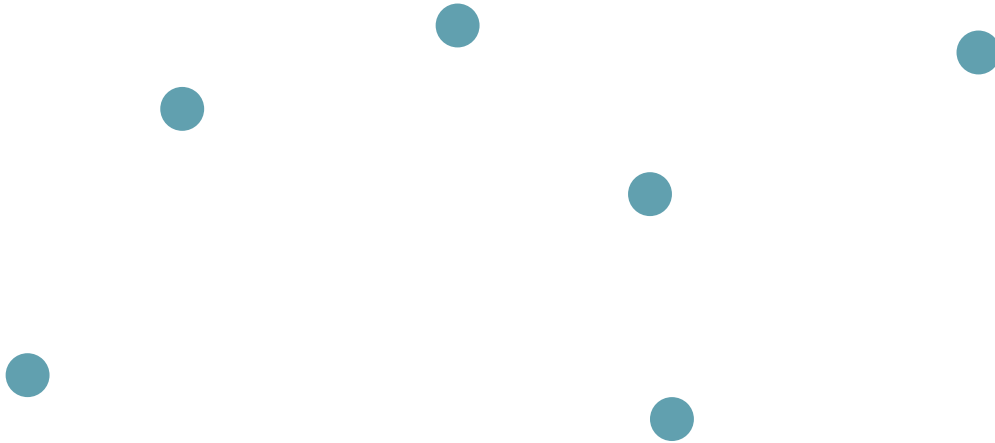


# Buffers

How many people live within walking distance of a grocery store?

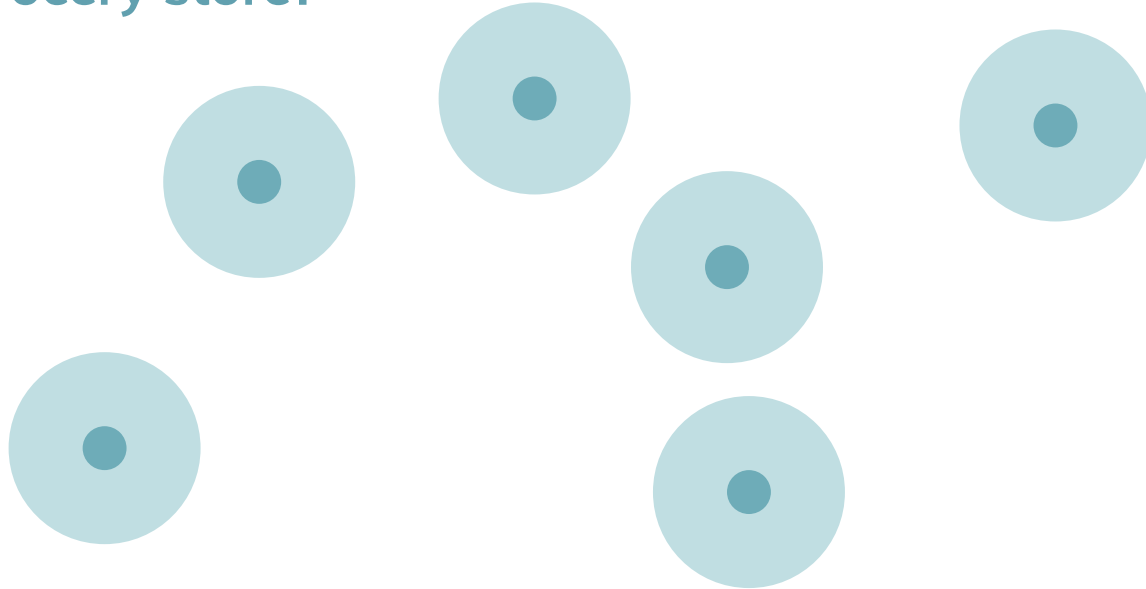
# Buffers

How many people live within walking distance of a grocery store?



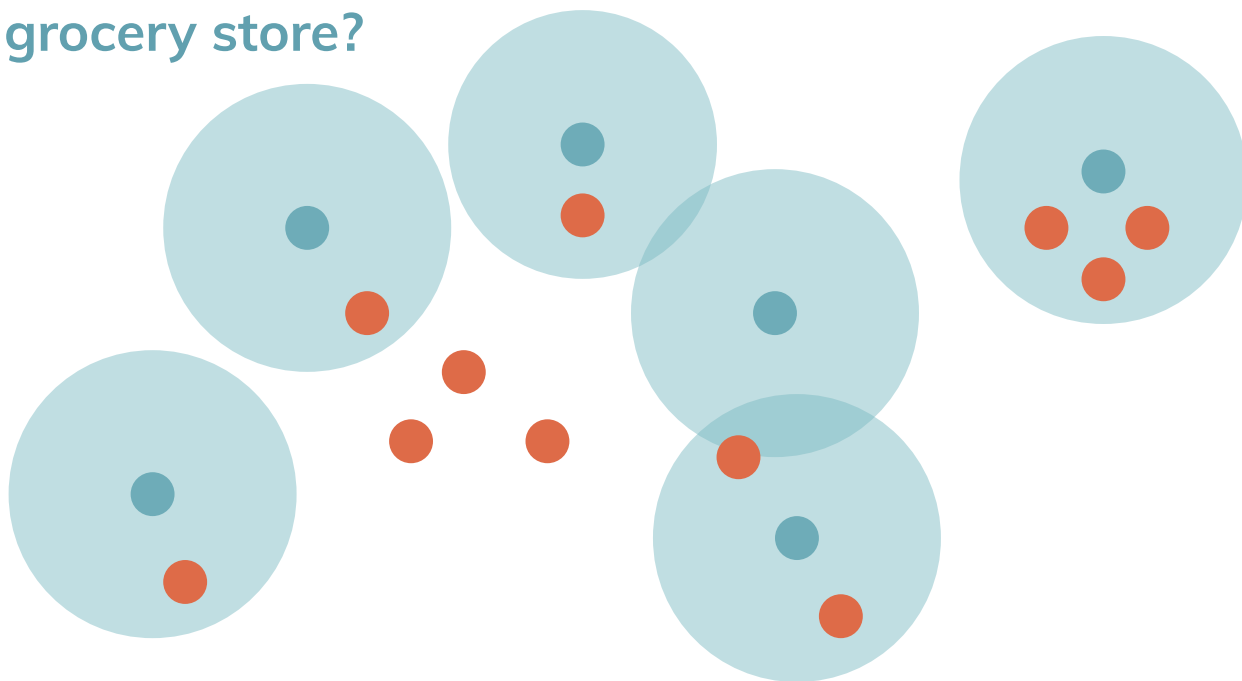
# Buffers

How many people live within walking distance of a grocery store?



# Buffers

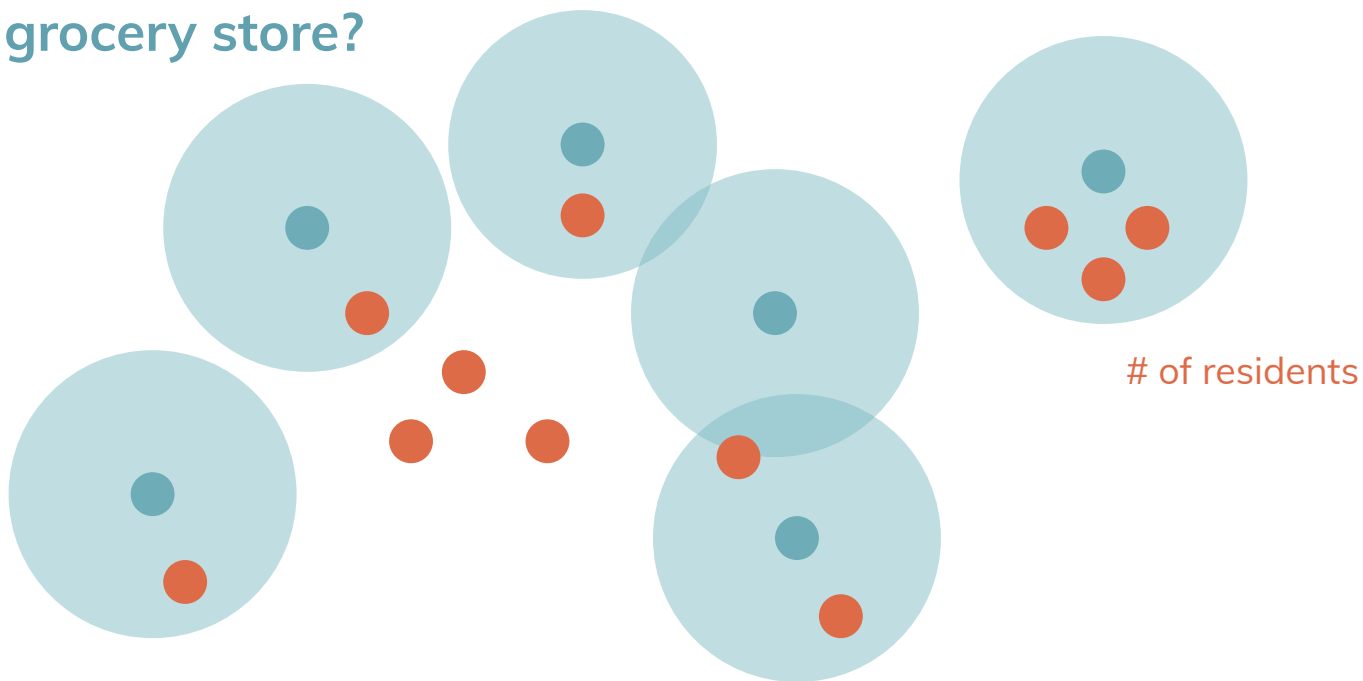
How many people live within walking distance of a grocery store?



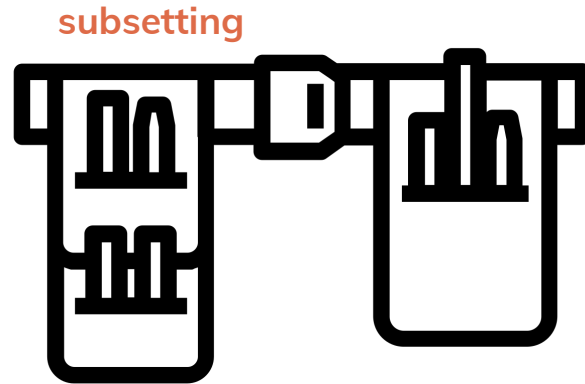


# Buffers

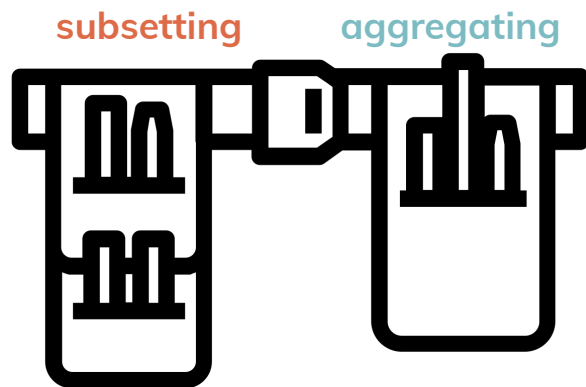
How many people live within walking distance of a grocery store?



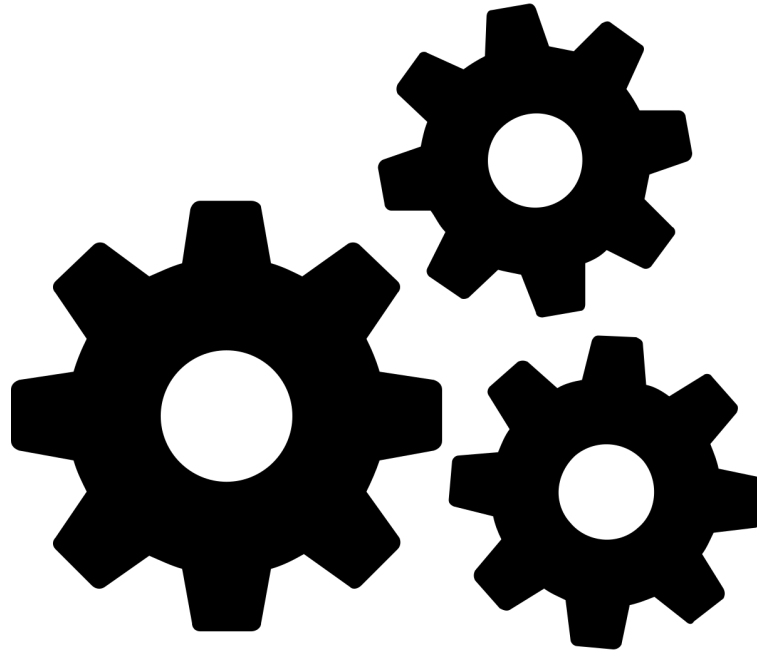
# Toolbelt for solving spatial problems



# Toolbelt for solving spatial problems



Switching gears...

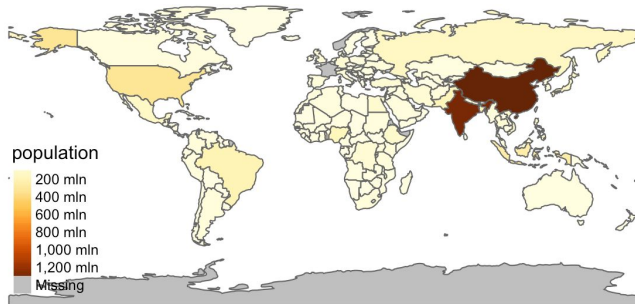


# Aggregation

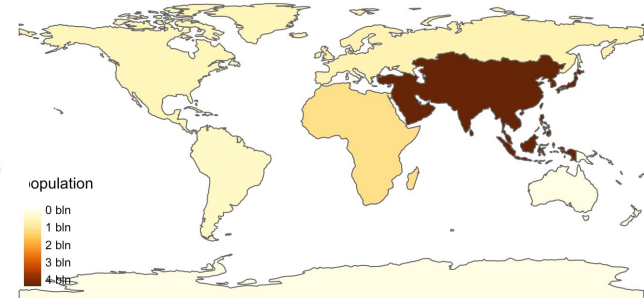
Which continent has the highest population?

# Aggregation

## Which continent has the highest population?



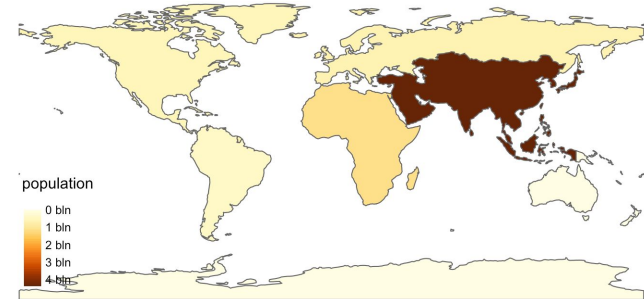
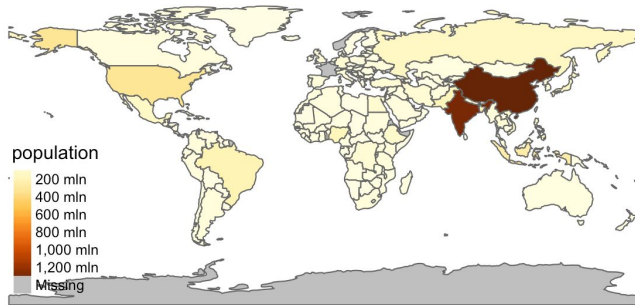
```
continents <- world %>%  
  group_by(continent) %>%  
  summarise(population = sum(pop, na.rm = TRUE))
```



Country	Continent
USA	North America
...	...

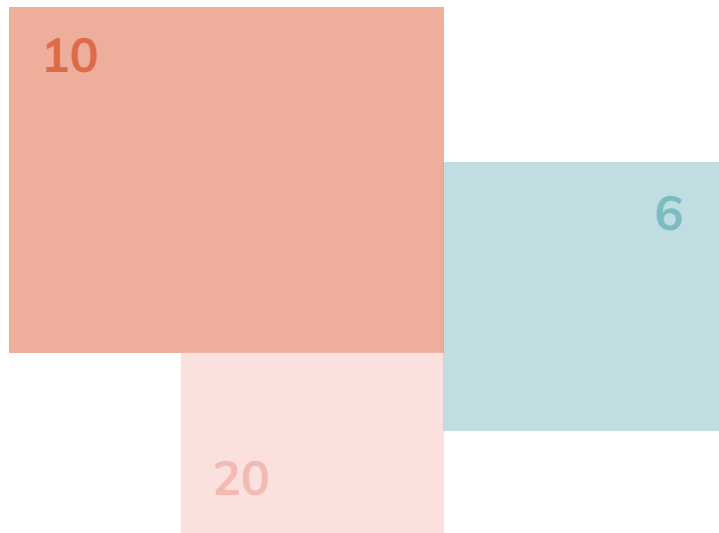
# Aggregation

Which continent has the highest population?



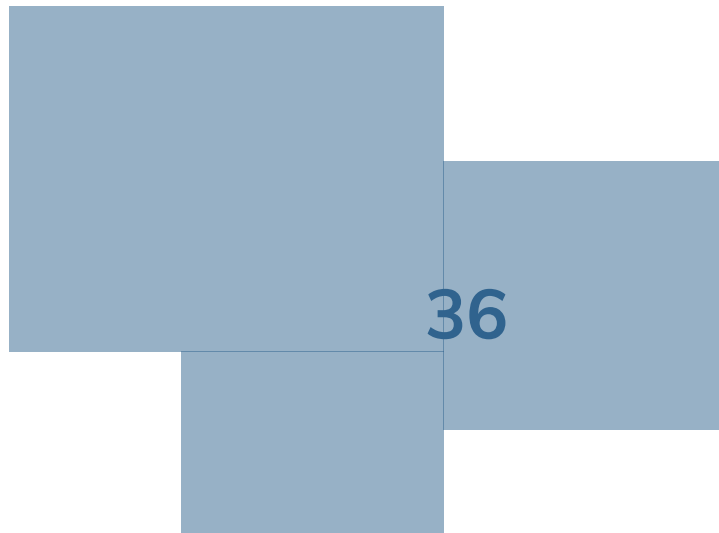
Country	Continent
USA	North America
...	...

# Geometry unions

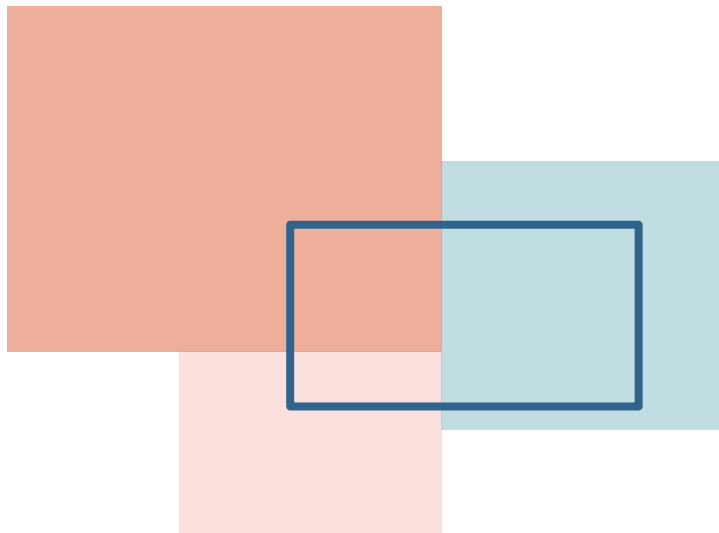




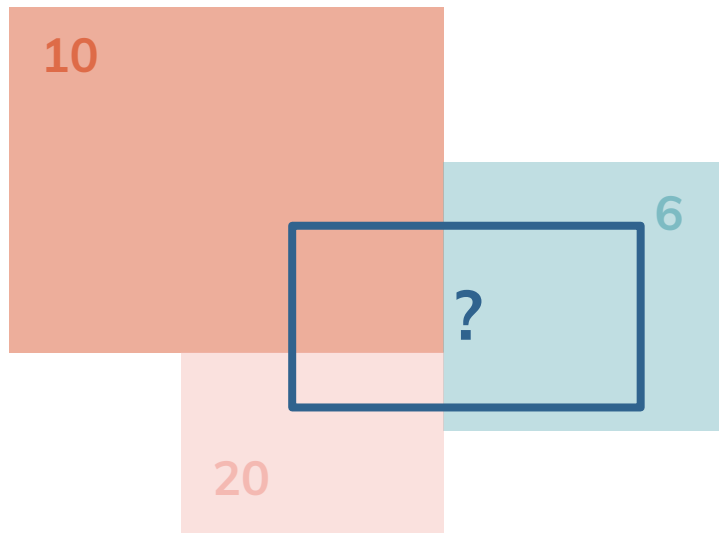
# Geometry unions



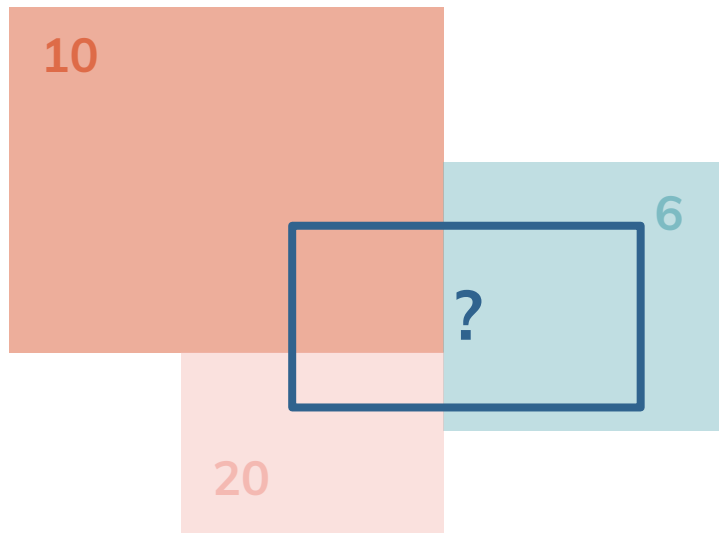
# Geometry unions



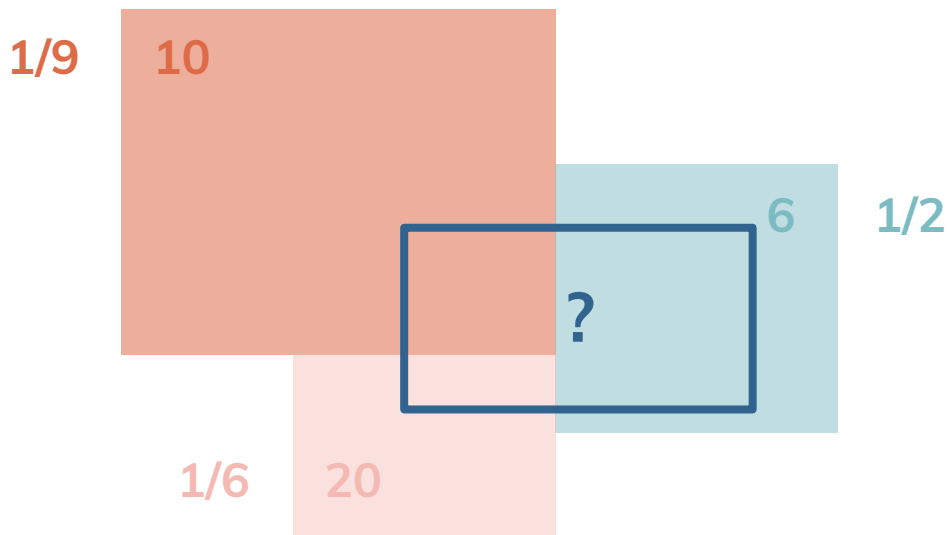
# Geometry unions



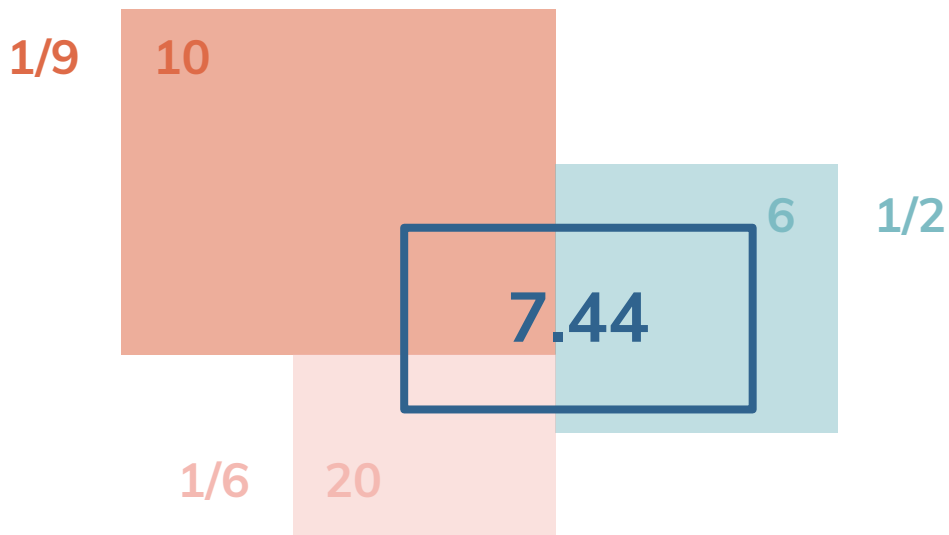
# Geometry unions



# Geometry unions: area-weighted interpolation

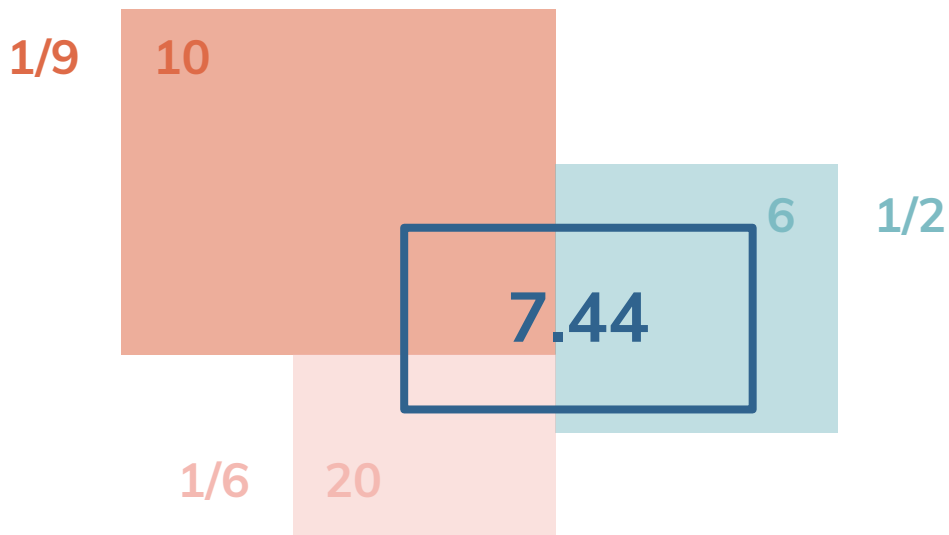


# Geometry unions: area-weighted interpolation

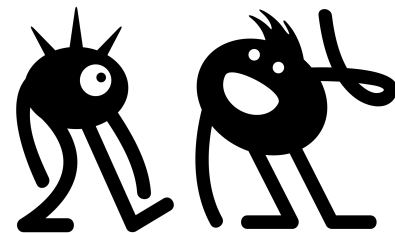


$$\frac{1}{9}(10) + \frac{1}{2}(6) + \frac{1}{6}(20) = 7.44$$

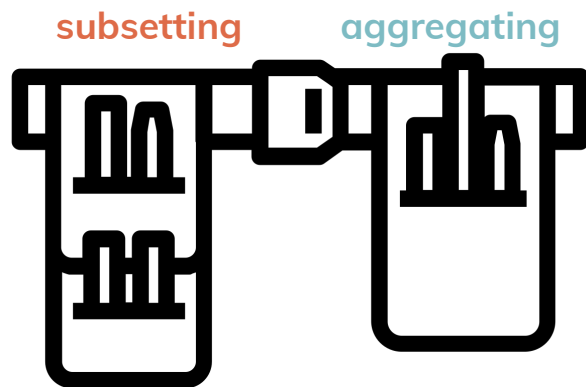
# Geometry unions: area-weighted interpolation



$$\frac{1}{9}(10) + \frac{1}{2}(6) + \frac{1}{6}(20) = 7.44$$

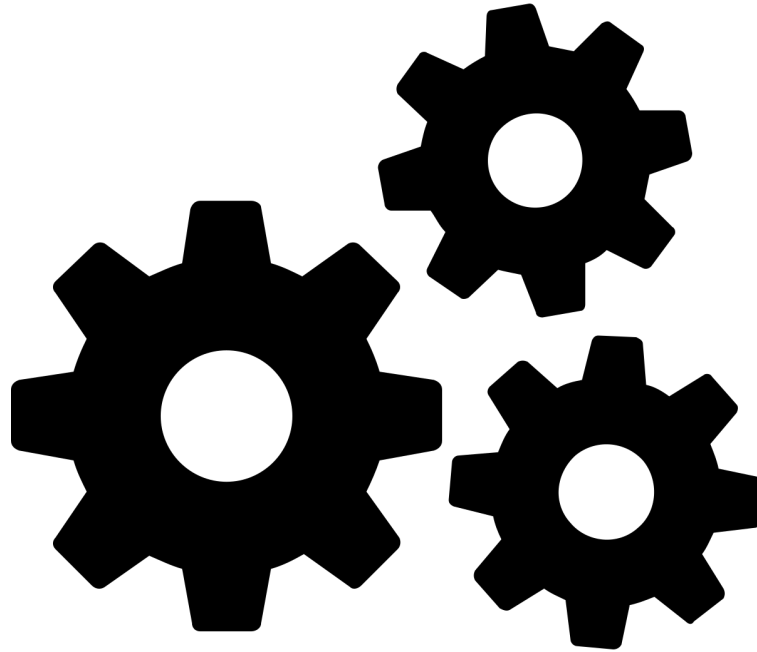


# Toolbelt for solving spatial problems

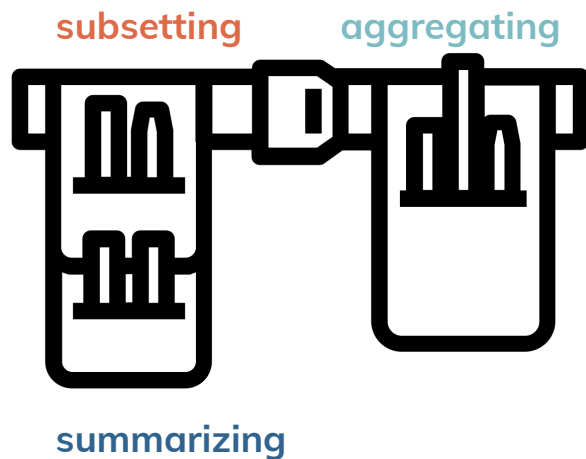




Switching gears...



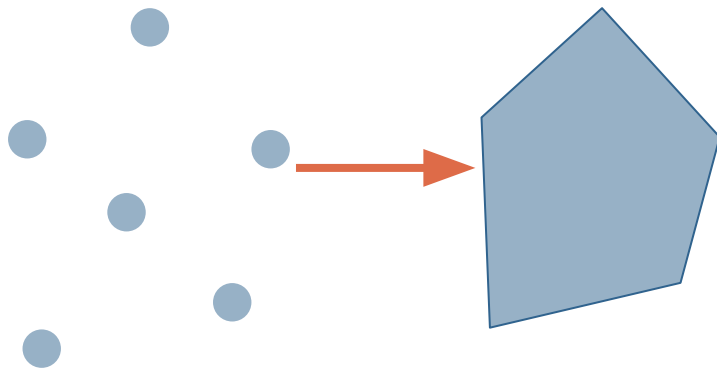
# Toolbelt for solving spatial problems



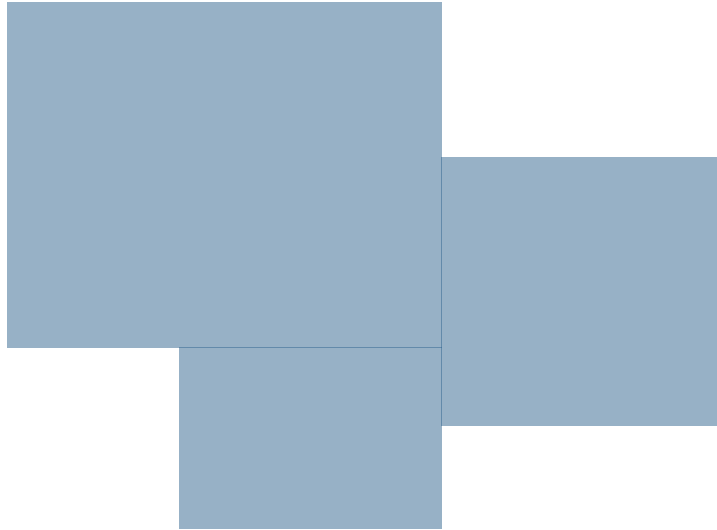
# Summarizing



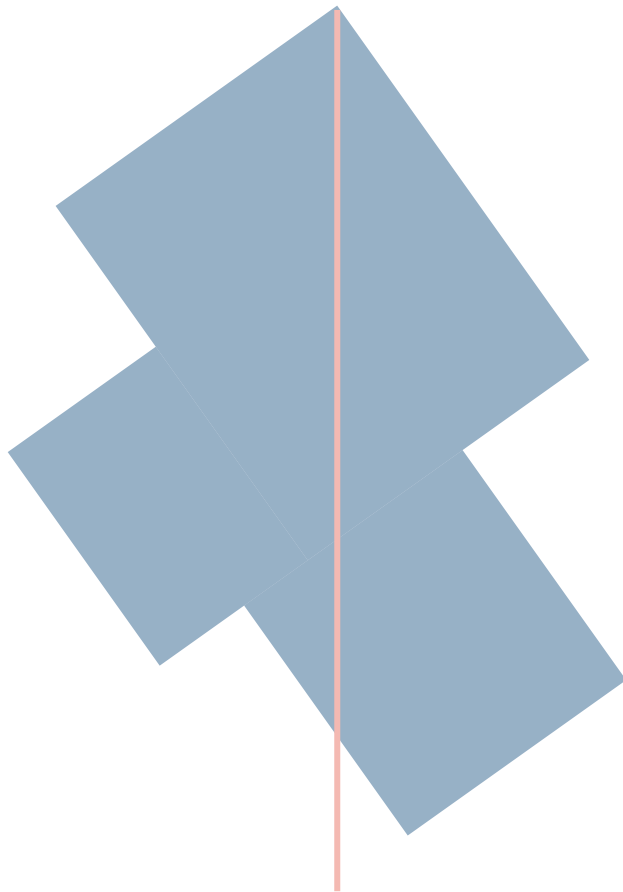
# Summarizing



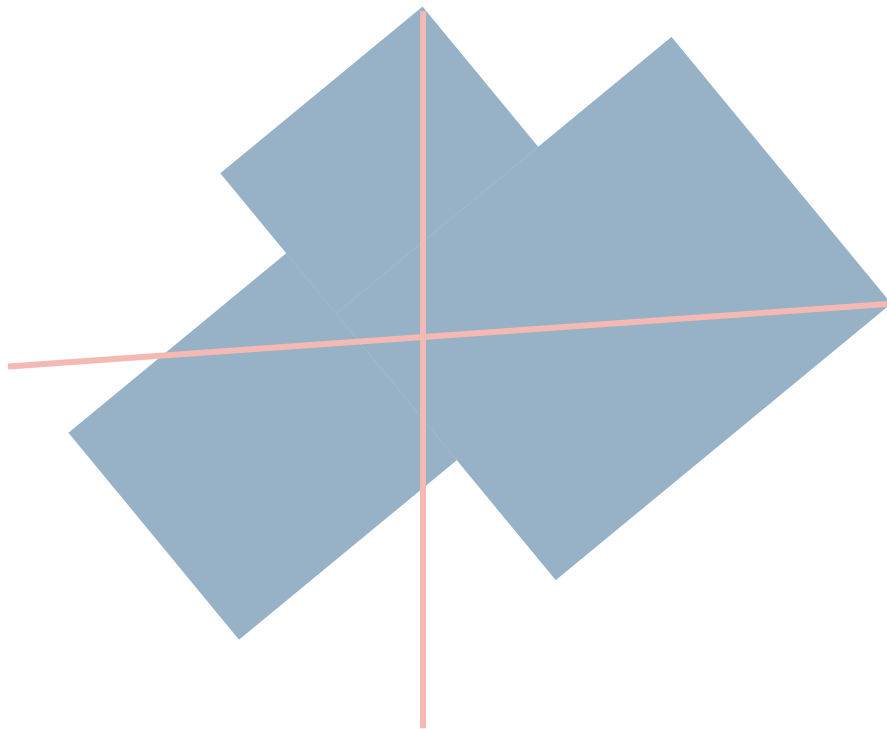
# Centroids



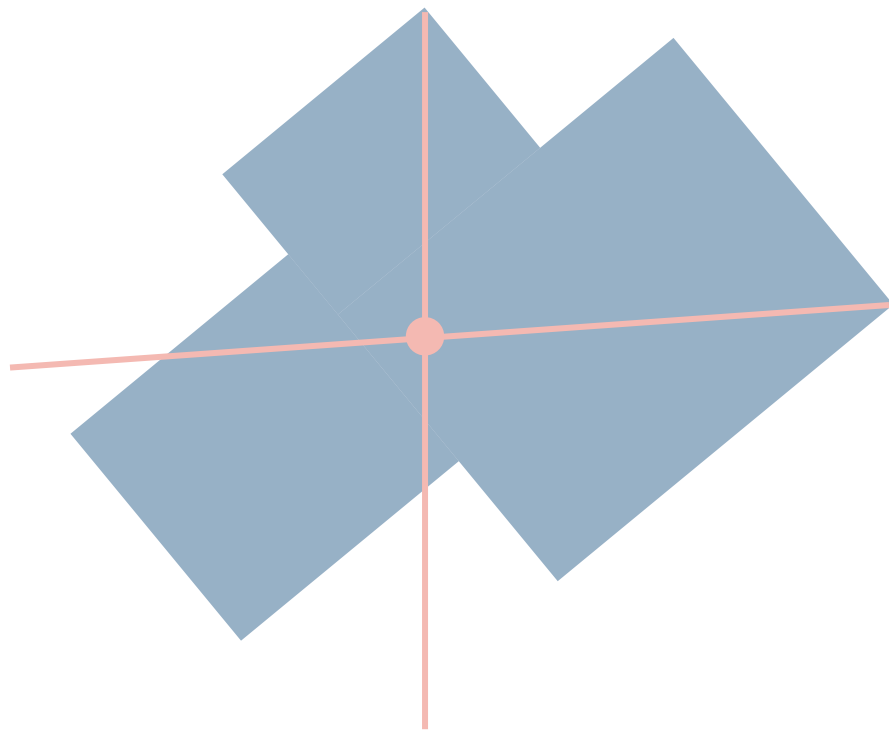
# Centroids



# Centroids

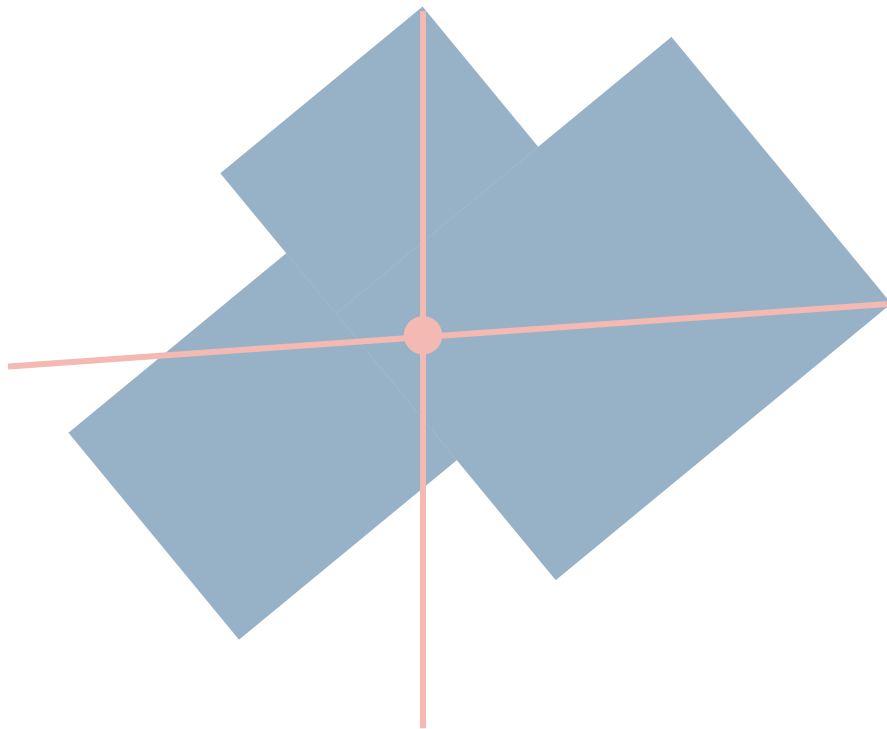


# Centroids

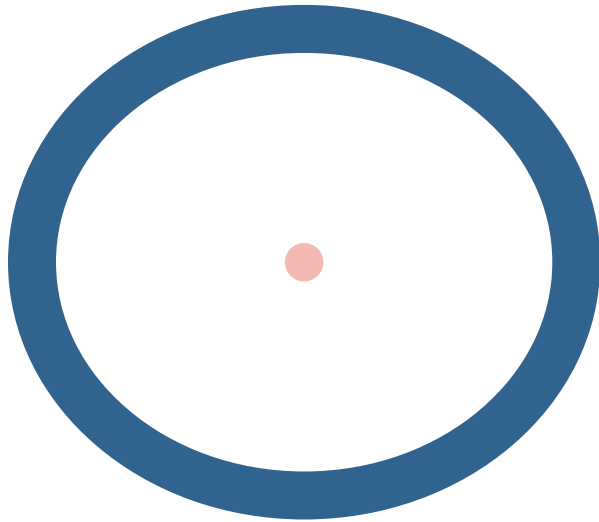
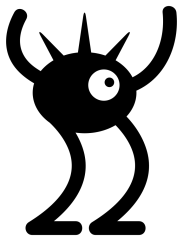




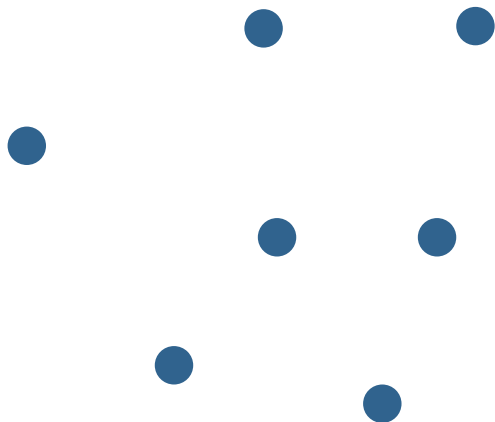
# Centroids



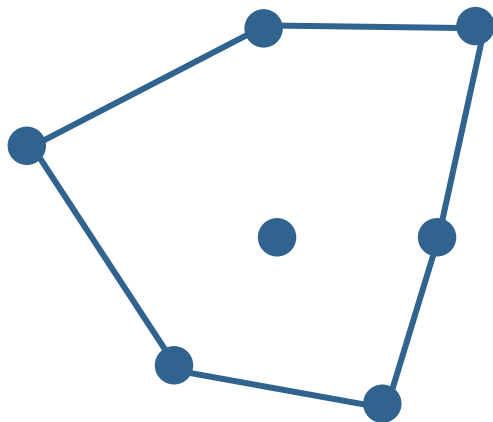
# Centroids



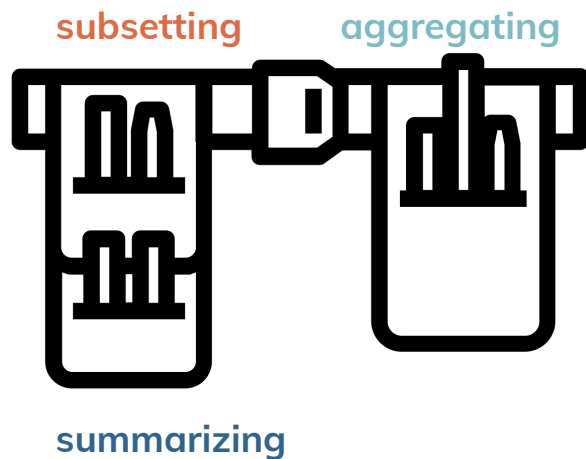
# Convex hulls



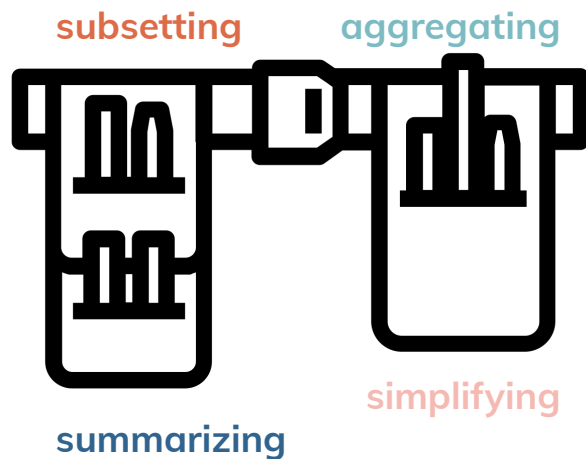
# Convex hulls



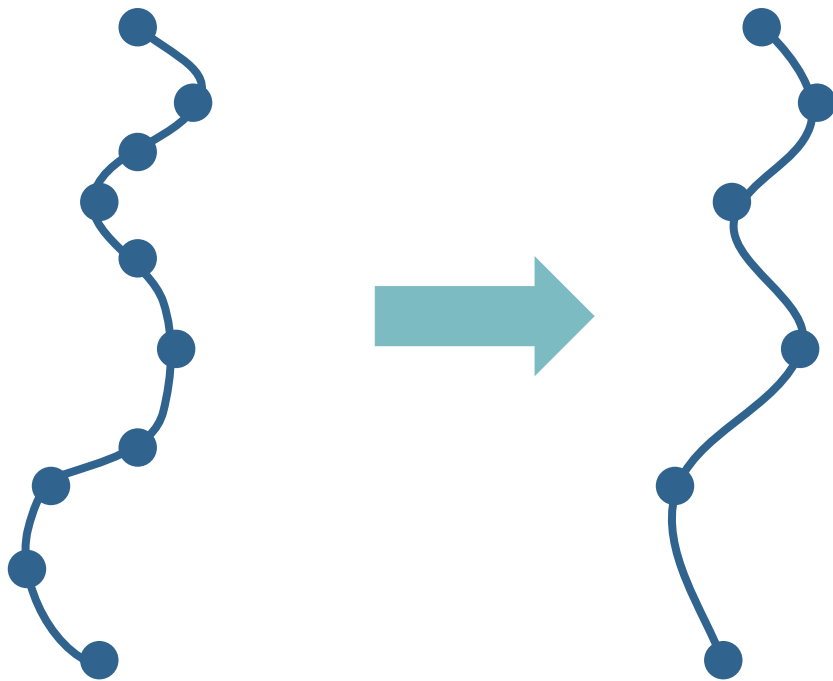
# Toolbelt for solving spatial problems



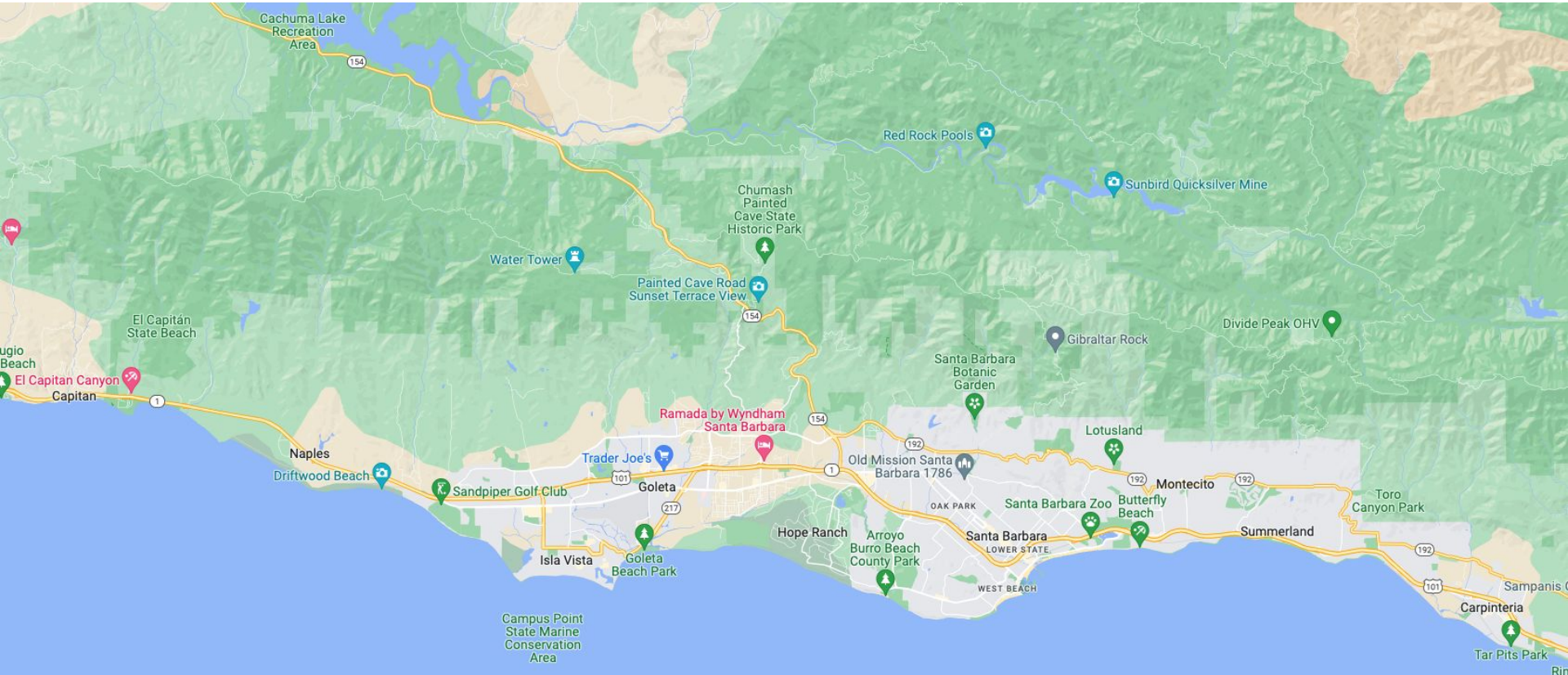
# Toolbelt for solving spatial problems



# Simplifications

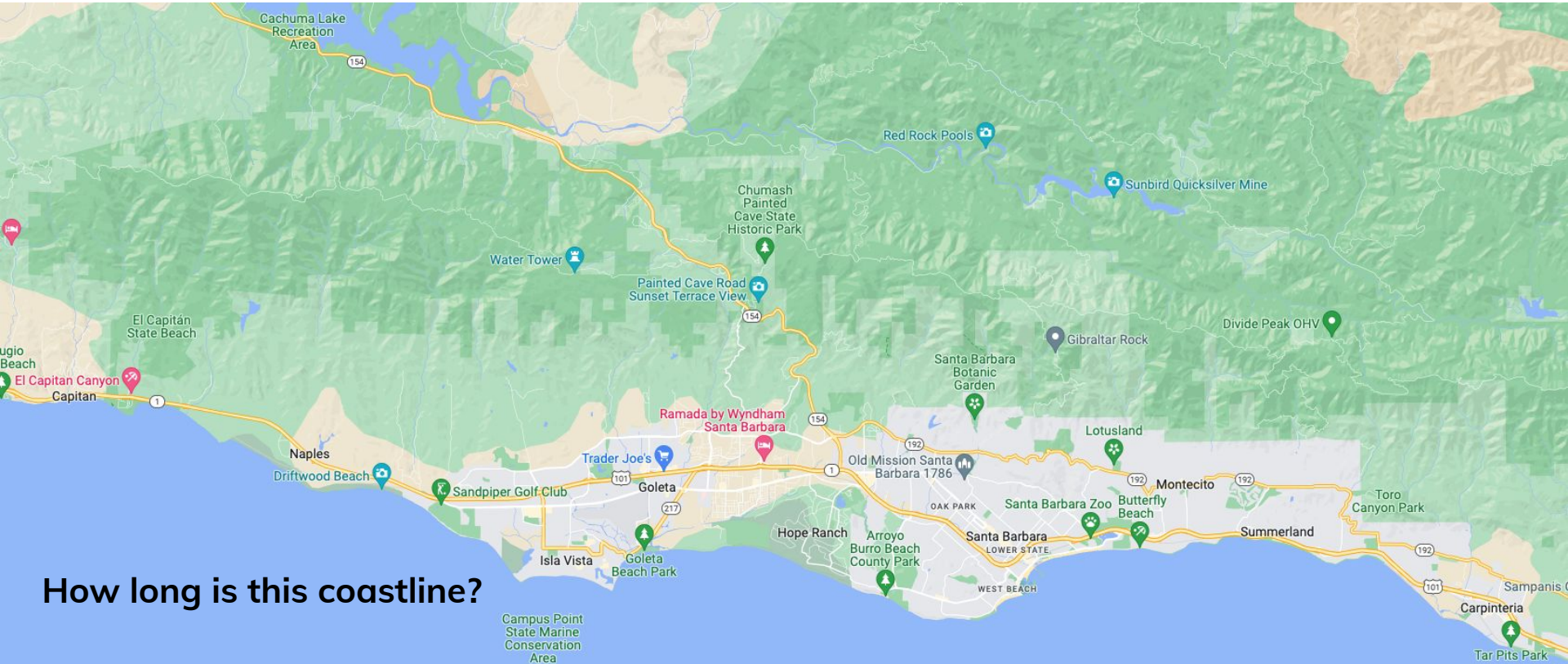


# Coastline paradox

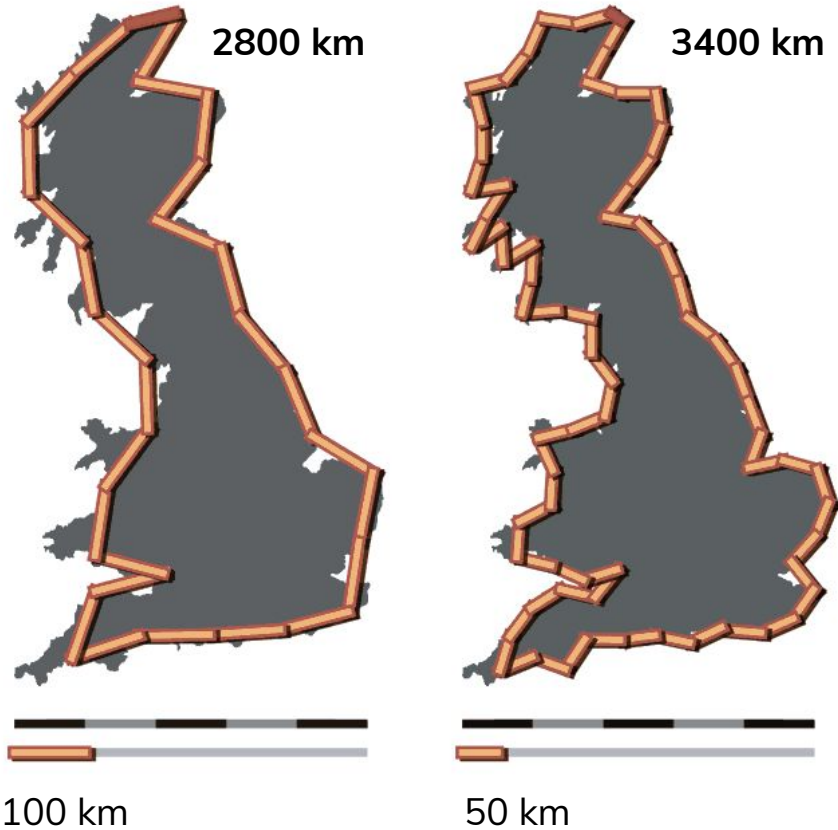




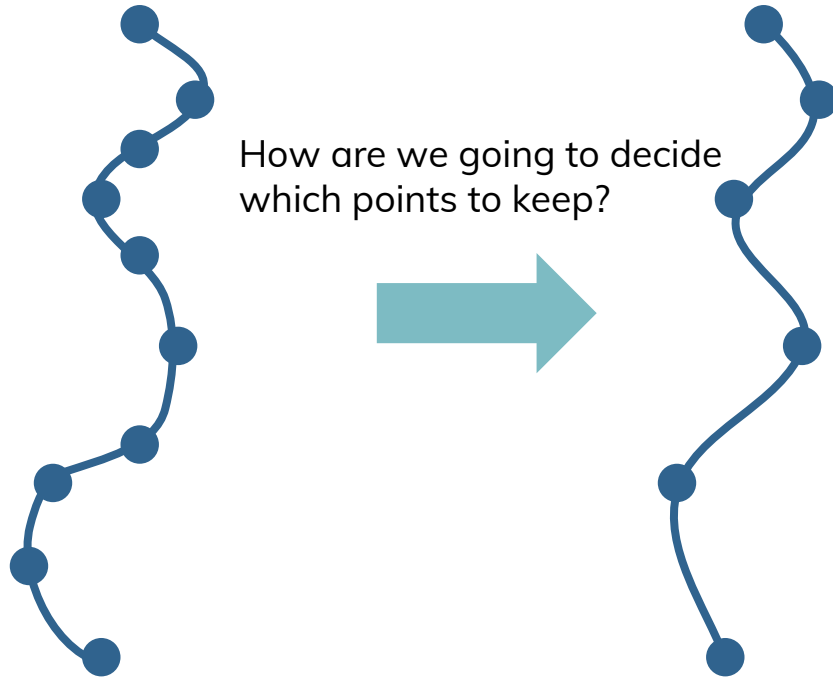
# Coastline paradox



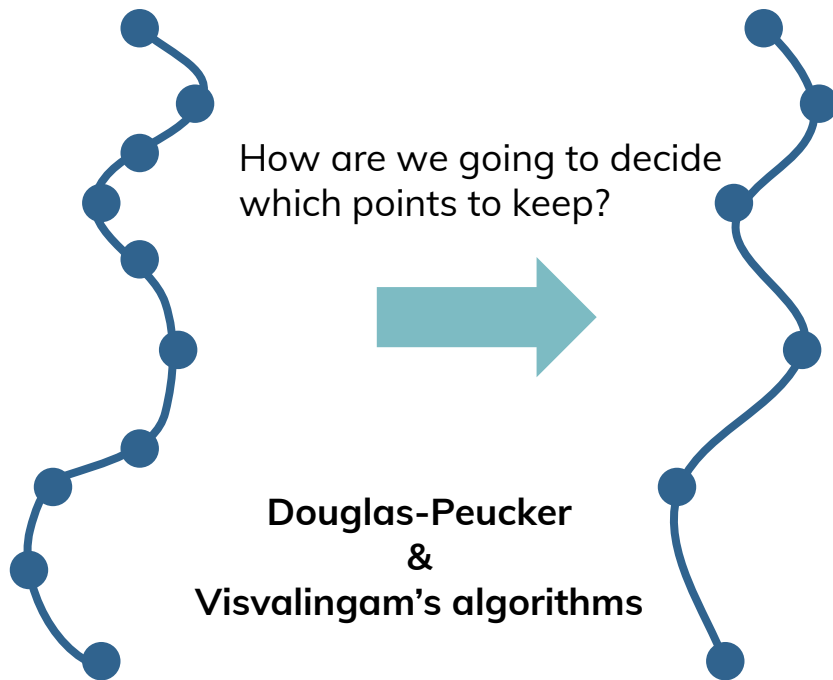
# Coastline paradox



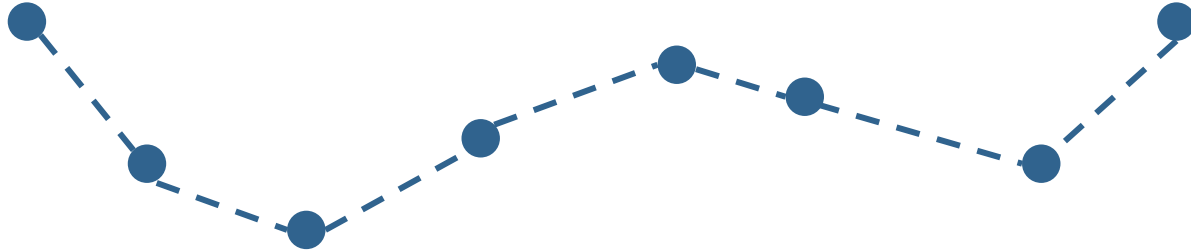
# Simplification



# Simplification

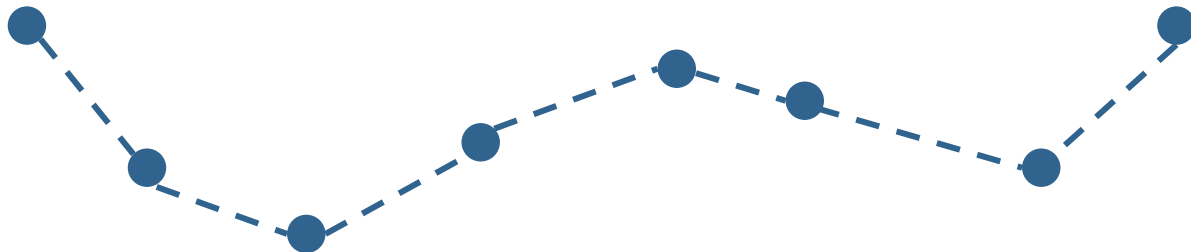


# Simplification: Douglas-Peucker algorithm



# Simplification: Douglas-Peucker algorithm

$$C = (P_1, P_2, P_3, \dots, P_n)$$

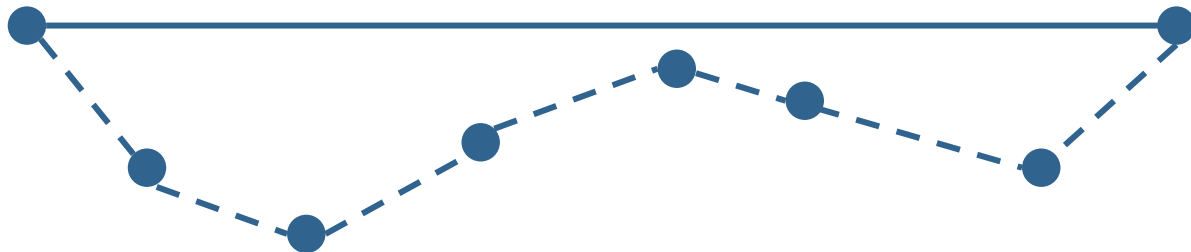


$\epsilon > 0$



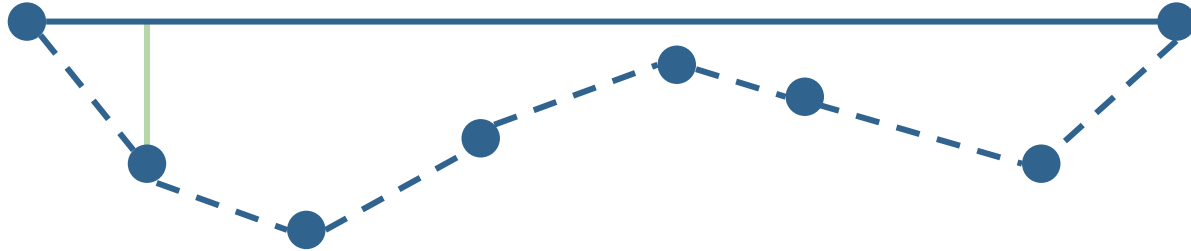
# Simplification: Douglas-Peucker algorithm

$\overline{P_1 P_n}$



# Simplification: Douglas-Peucker algorithm

$$\overline{P_1 P_n}$$

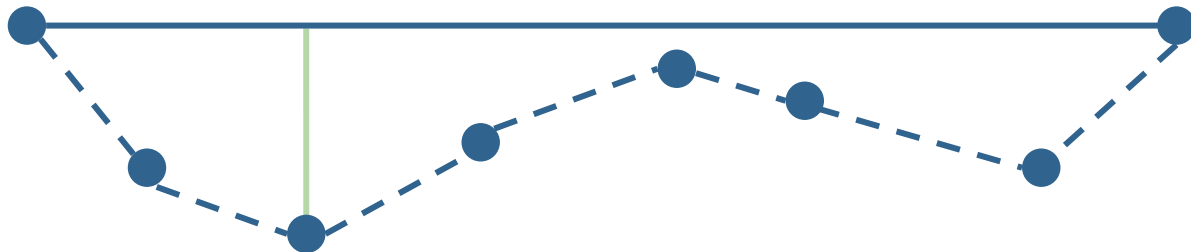


$$d(P_1, \overline{P_1 P_n})$$



# Simplification: Douglas-Peucker algorithm

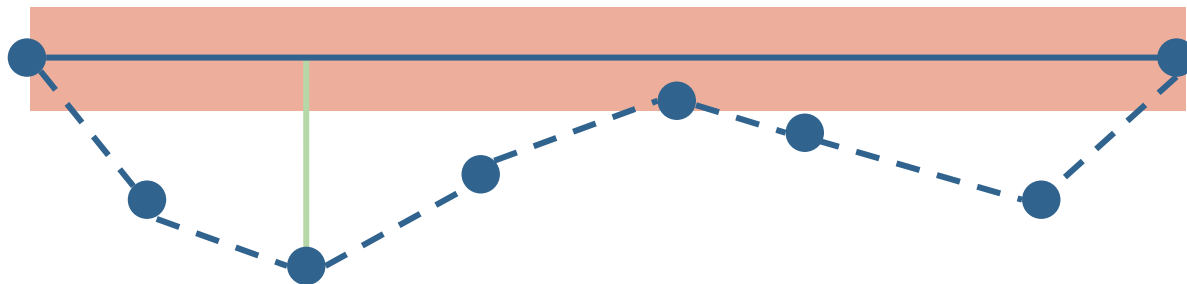
$\overline{P_1 P_n}$



$$d_{max} = \max_{i=2 \dots n-1} d(P_i, \overline{P_1 P_n})$$

# Simplification: Douglas-Peucker algorithm

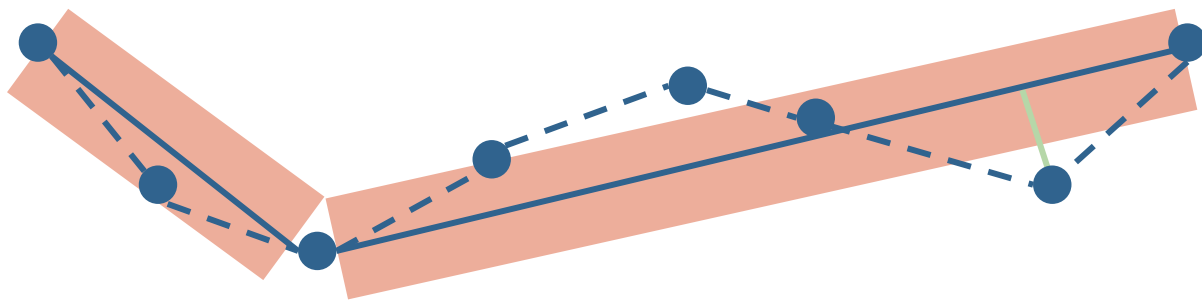
$\overline{P_1 P_n}$



$$d_{max} = \max_{i=2 \dots n-1} d(P_i, \overline{P_1 P_n}) \leq \epsilon$$

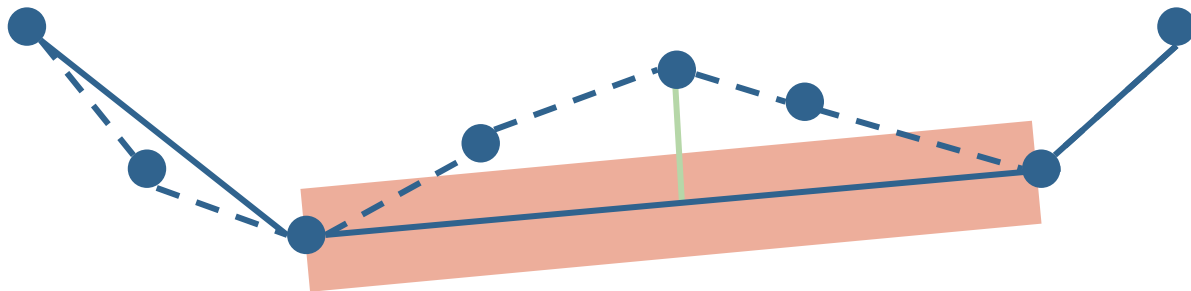
# Simplification: Douglas-Peucker algorithm

$\overline{P_1 P_m}$     $\overline{P_m P_n}$



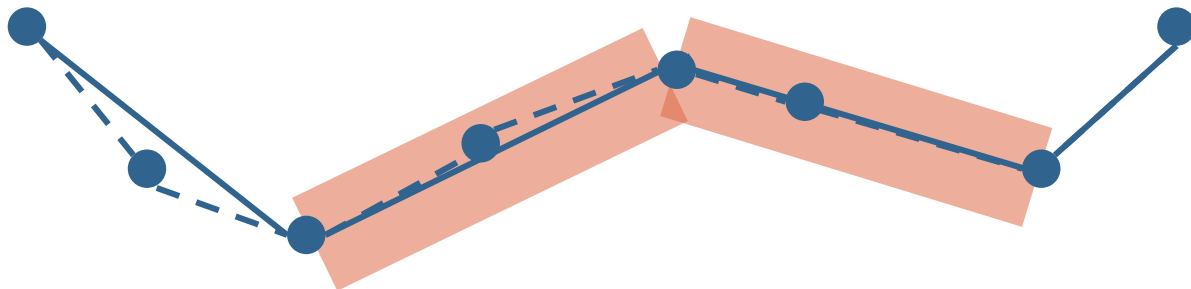
$$d_{max} = \max_{i=2 \dots n-1} d(P_i, \overline{P_1 P_n}) \leq \epsilon$$

# Simplification: Douglas-Peucker algorithm

 $\overline{P_1 P_m}$  $\overline{P_m P_n}$ 

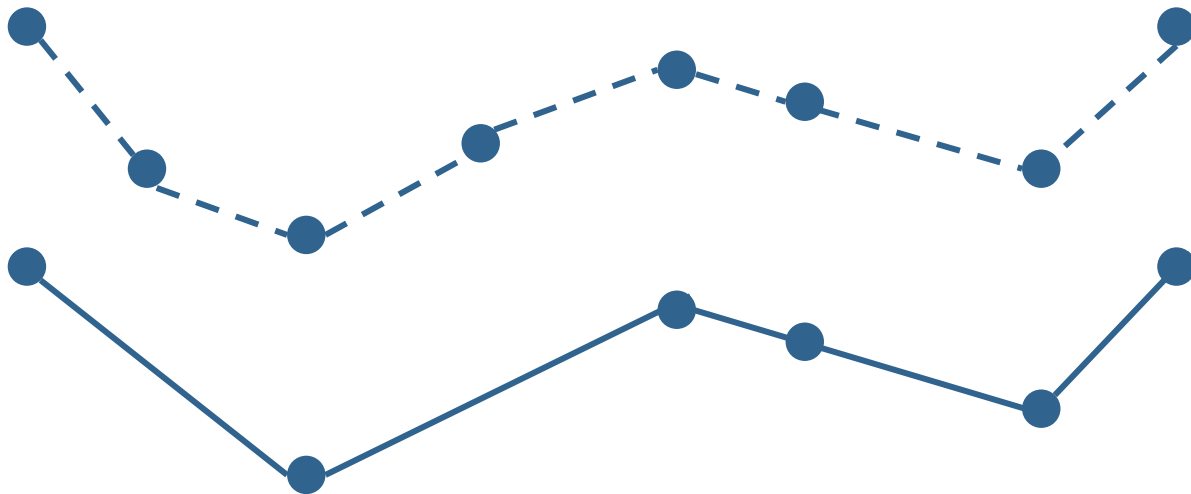
$$d_{max} = \max_{i=2 \dots n-1} d(P_i, \overline{P_1 P_n}) \leq \epsilon$$

# Simplification: Douglas-Peucker algorithm

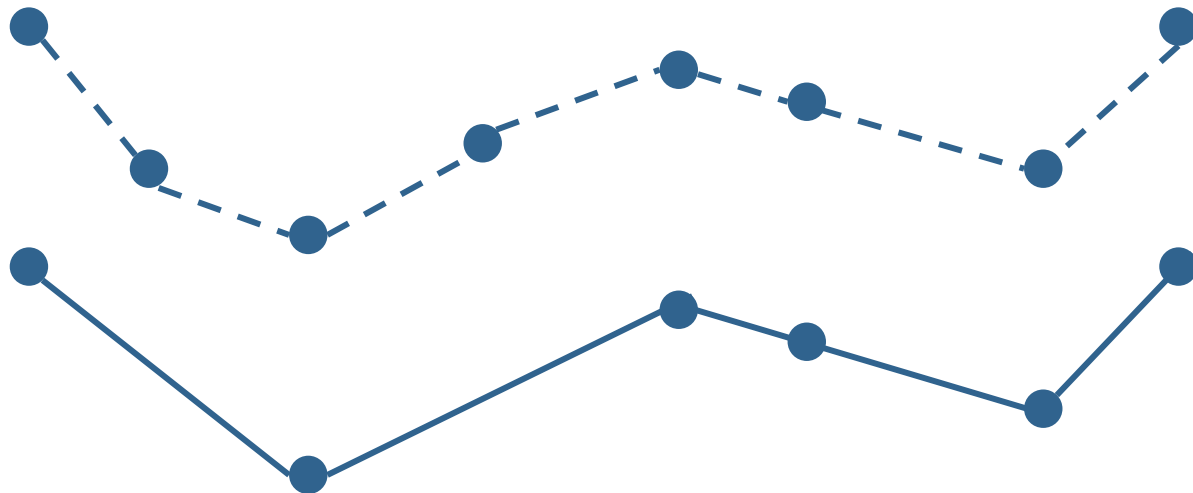
 $\overline{P_1 P_m}$  $\overline{P_m P_n}$ 

$$d_{max} = \max_{i=2 \dots n-1} d(P_i, \overline{P_1 P_n}) \leq \epsilon$$

# Simplification: Douglas-Peucker algorithm



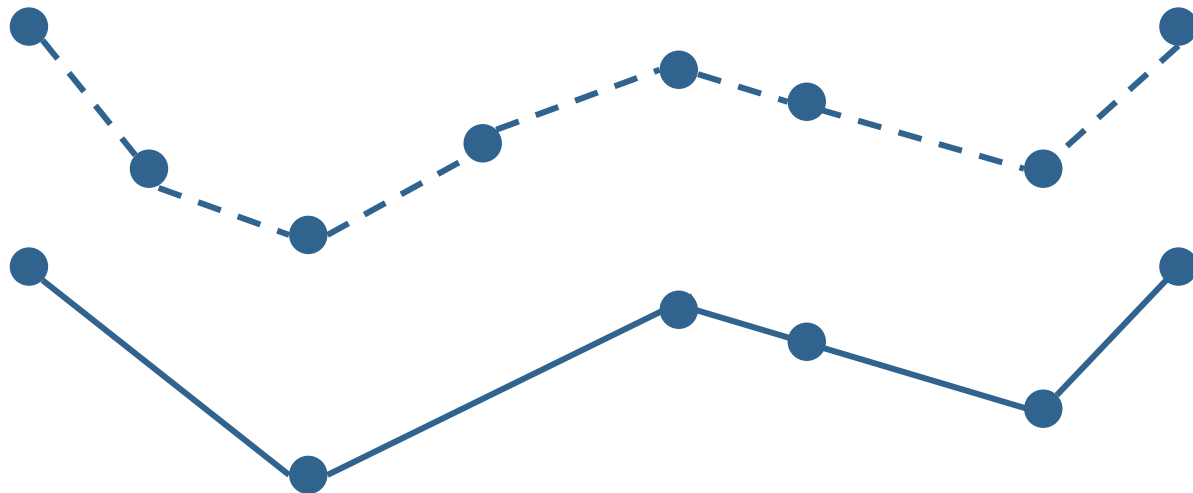
# Simplification: Douglas-Peucker algorithm



$\epsilon > 0$



# Simplification: Douglas-Peucker algorithm



$\epsilon > 0$





# Simplification: Douglas-Peucker algorithm

 CARTOGRAPHY PLAYGROUND

## Hands-On

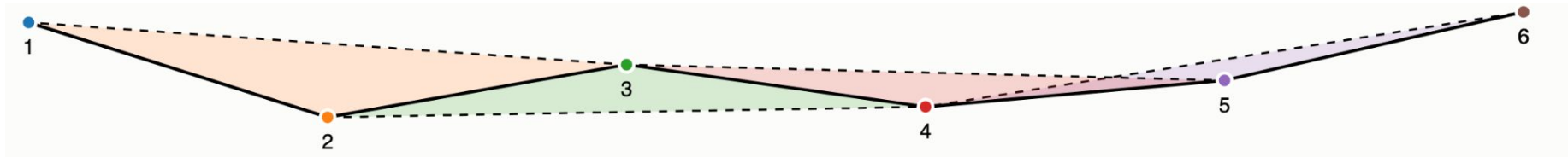
Drag the Slider to change the value of  $\epsilon$  and simplify the drawn curve. Click in the grid to draw new points. You can clear the curve by clicking on the *Clear* Button or you can restore a default path by choosing one from the *Reset* Button. The original line is displayed in dashed gray and the simplified line is displayed in solid blue.



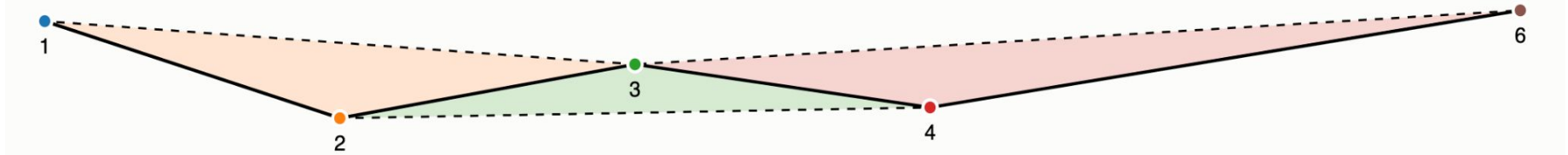
# Simplification: Visvalingam's algorithm



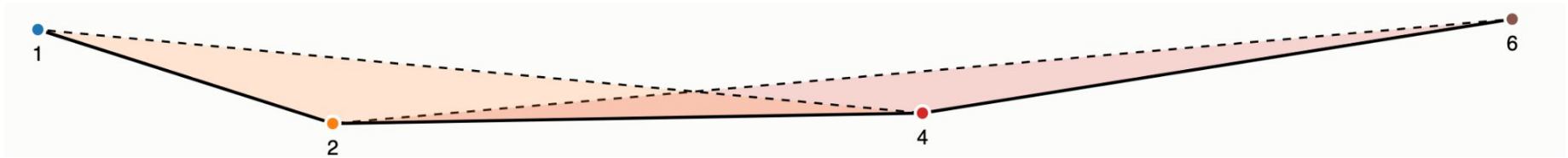
# Simplification: Visvalingam's algorithm



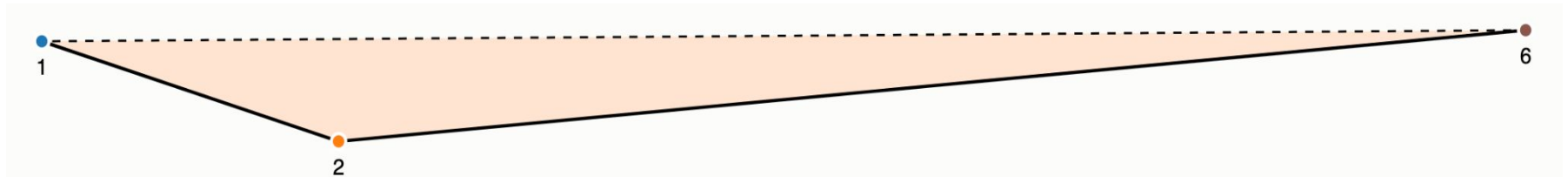
# Simplification: Visvalingam's algorithm



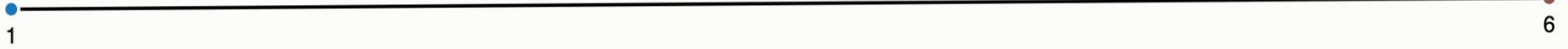
# Simplification: Visvalingam's algorithm



# Simplification: Visvalingam's algorithm



# Simplification: Visvalingam's algorithm



# Simplification: Visvalingam's algorithm

June 1, 2012 / Mike Bostock

## Line Simplification





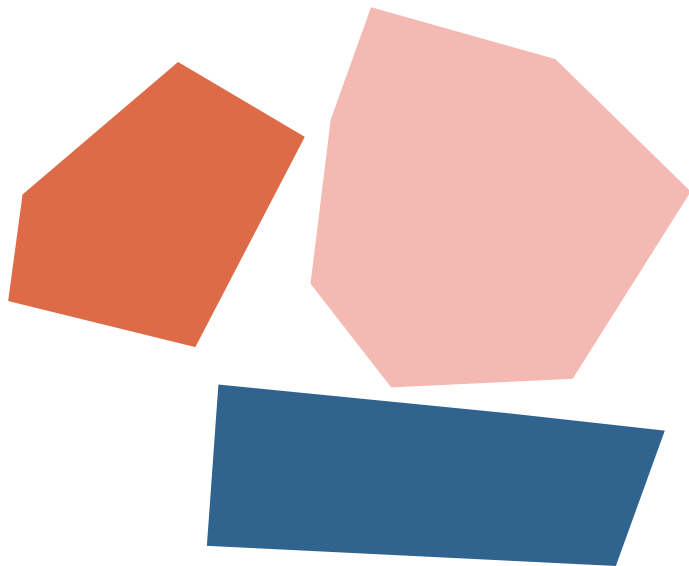
# Simplification



# Simplification



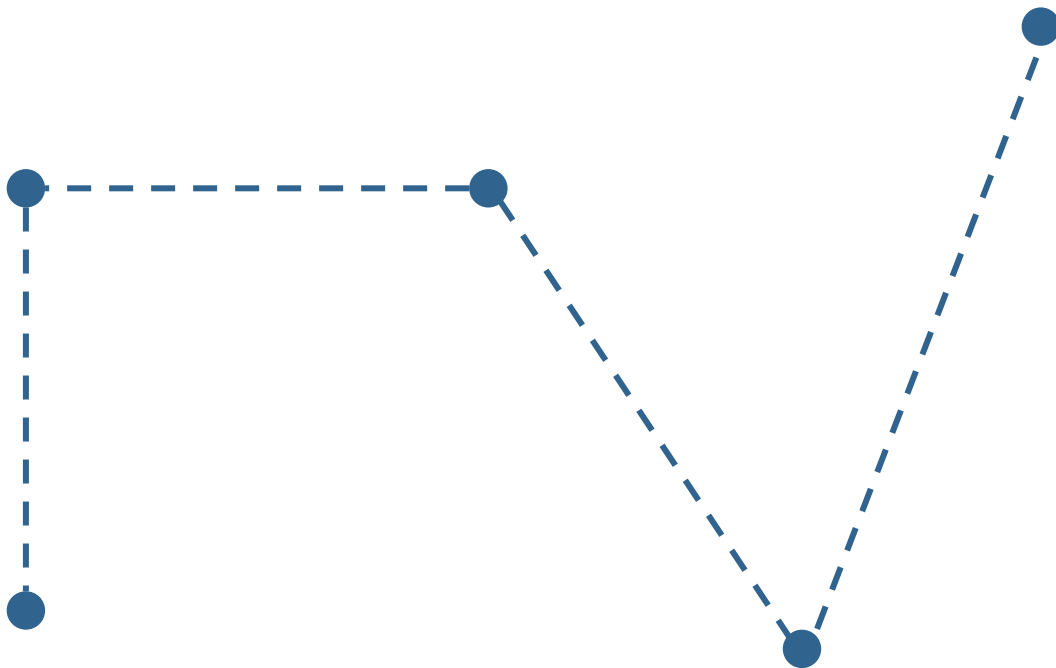
# Simplification



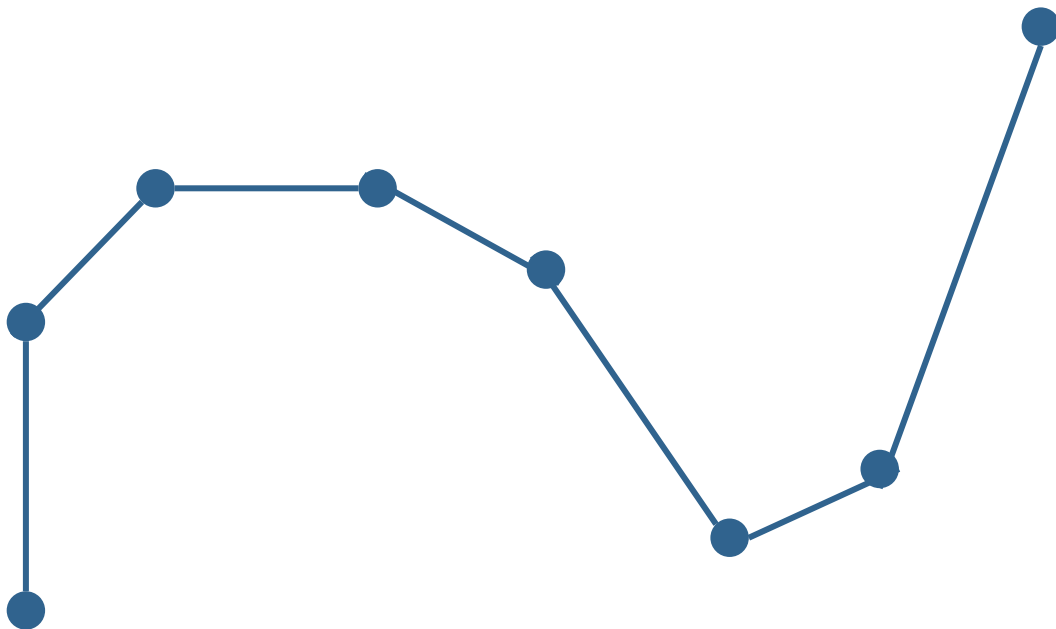
# Smoothing



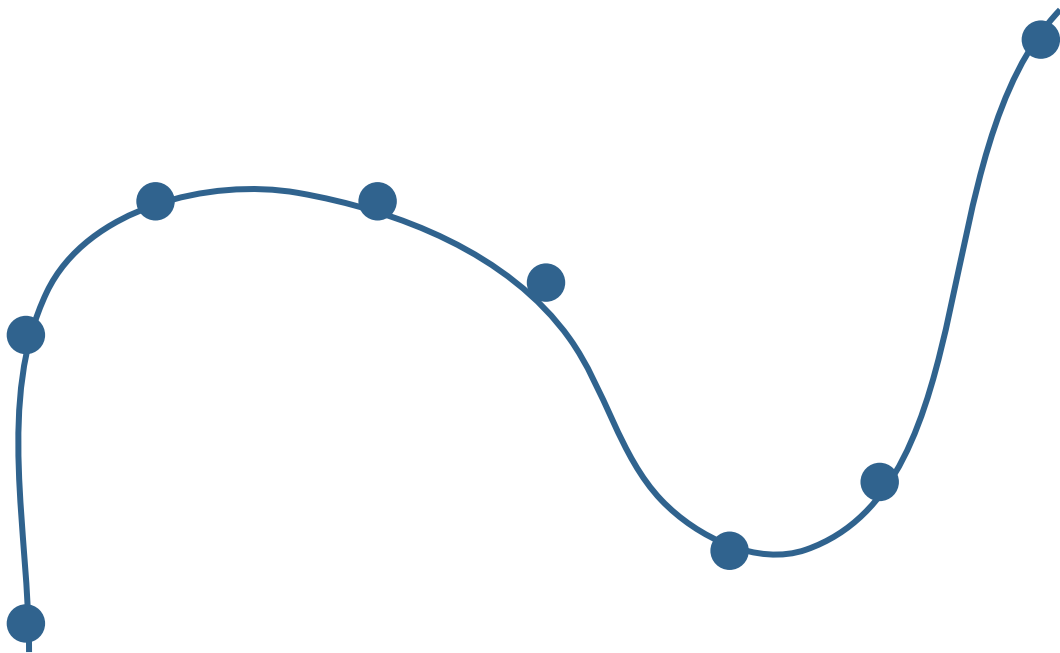
# Smoothing: Chaikin's corner cutting algorithm



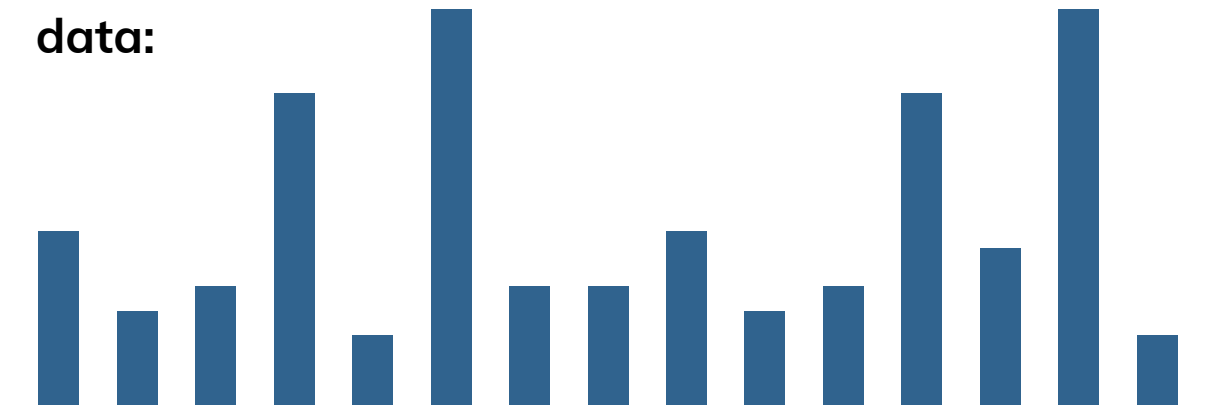
# Smoothing: Chaikin's corner cutting algorithm



# Smoothing: Chaikin's corner cutting algorithm



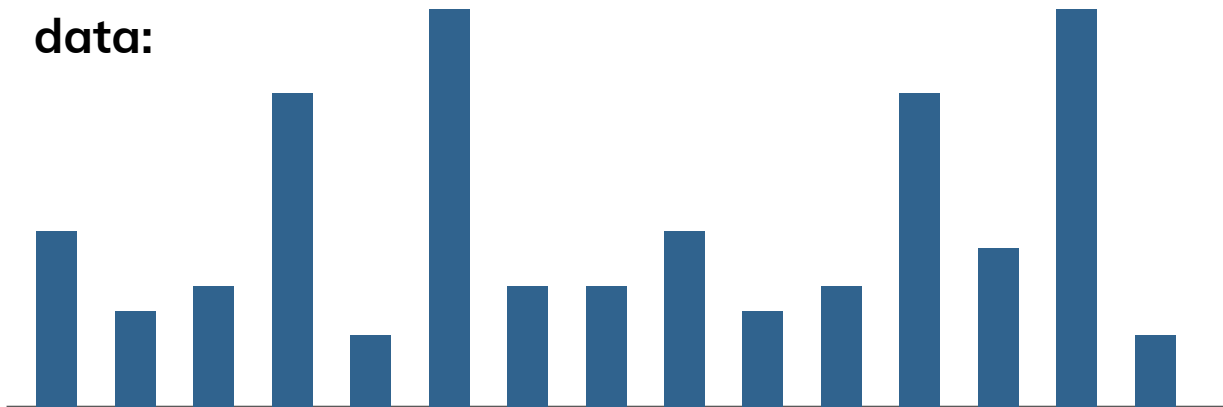
# Smoothing: Gaussian kernel



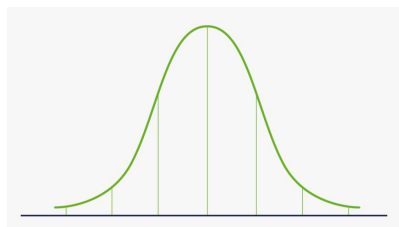


# Smoothing: Gaussian kernel

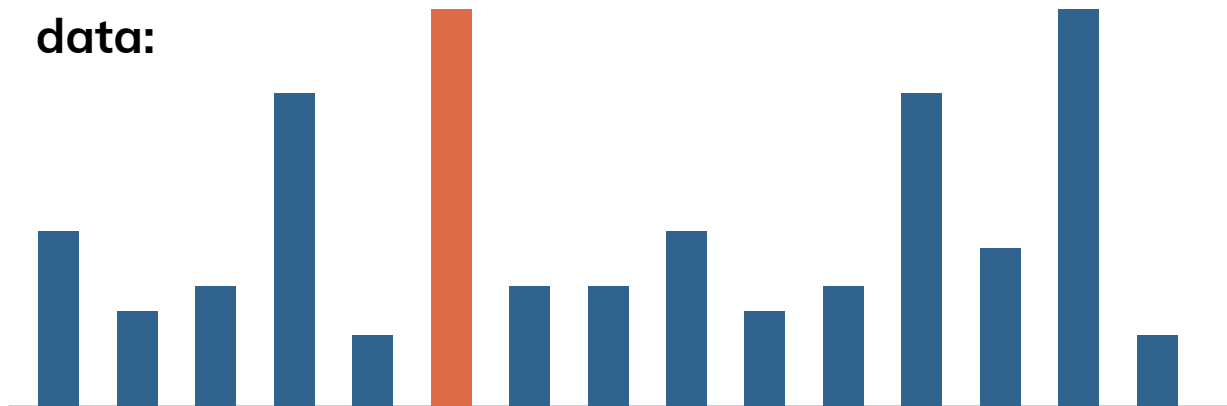
**data:**



**Gaussian distribution:**

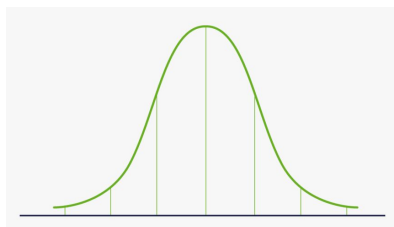
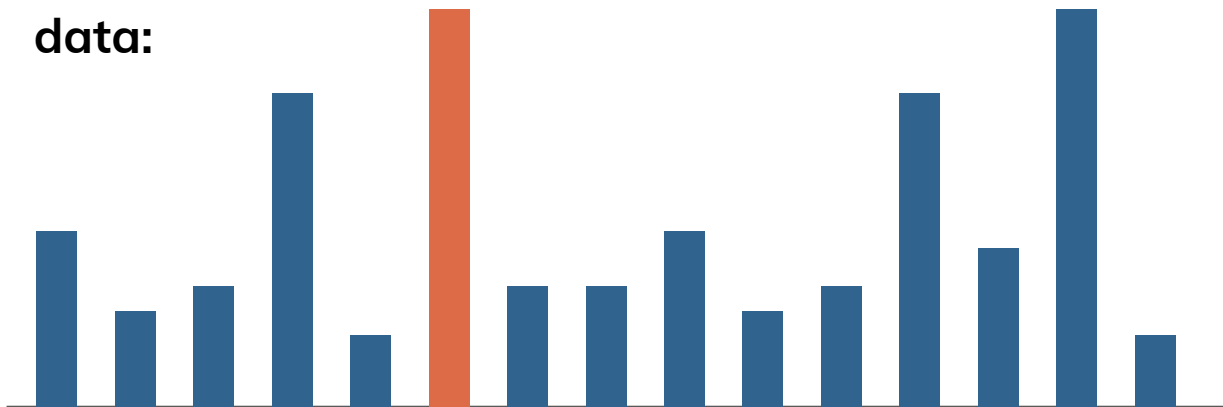


# Smoothing: Gaussian kernel

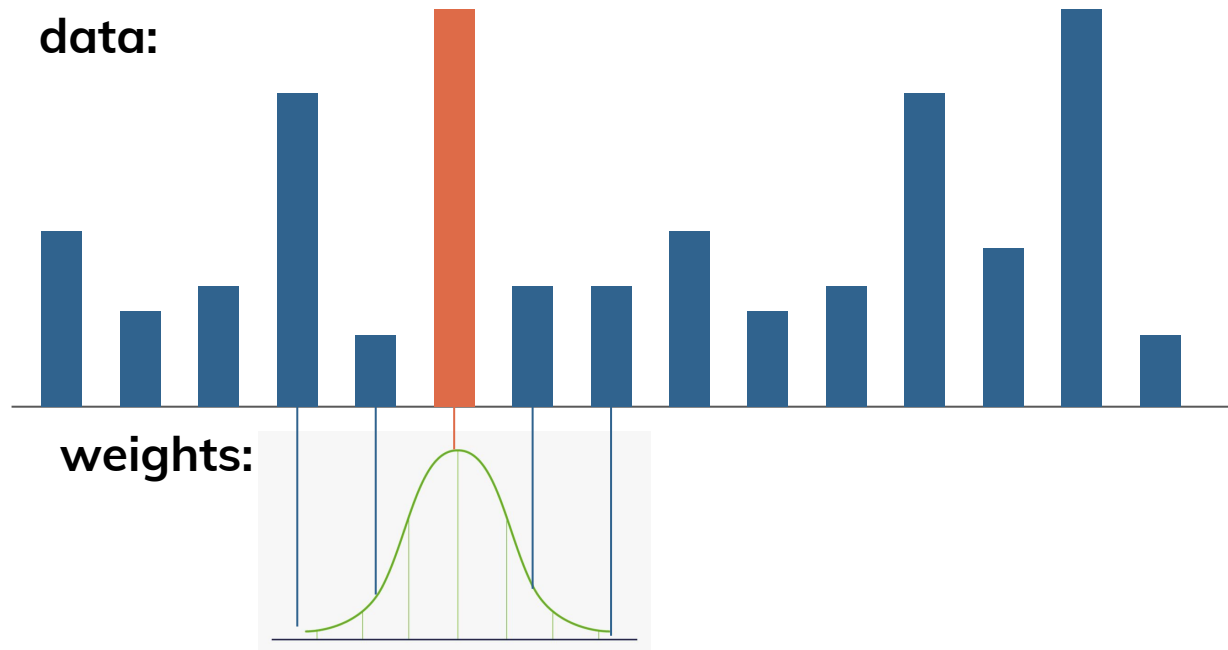


# Smoothing: Gaussian kernel

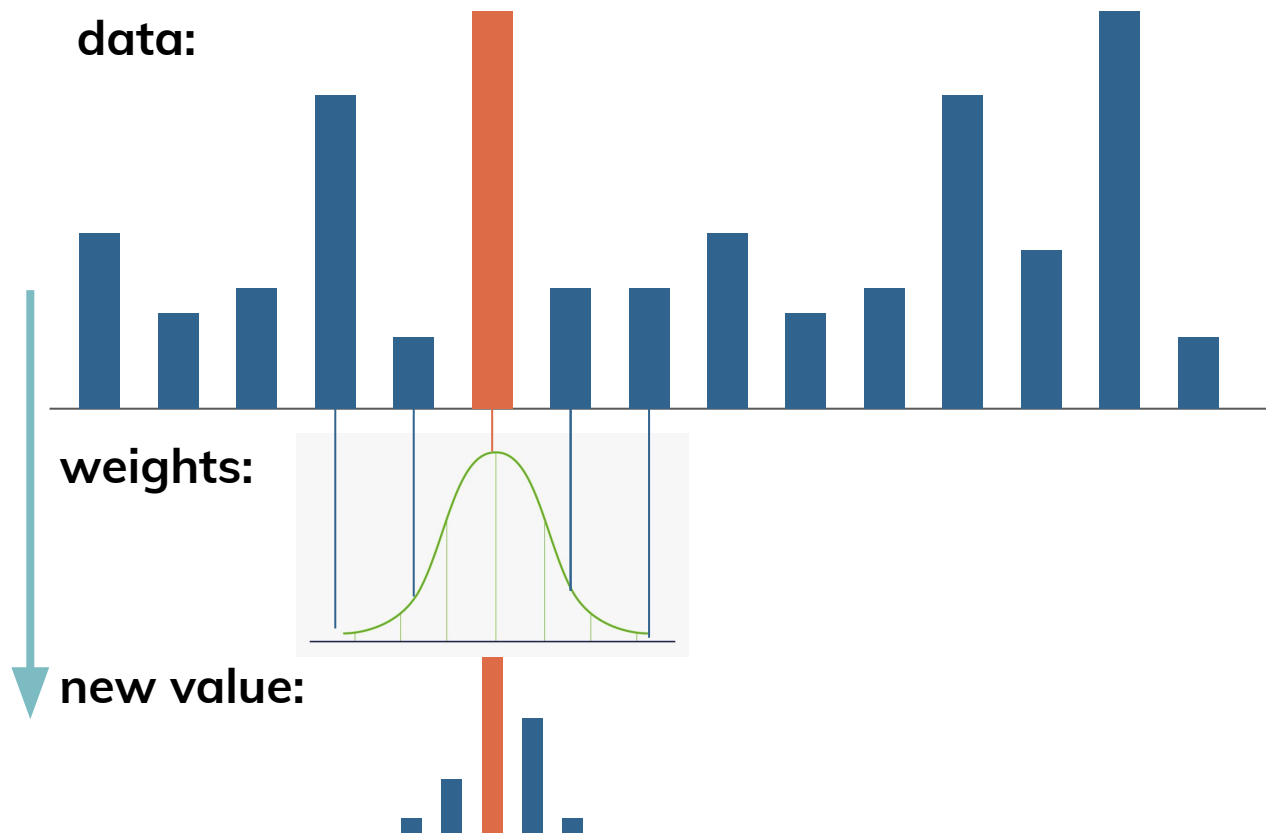
data:



# Smoothing: Gaussian kernel



# Smoothing: Gaussian kernel



# Toolbelt for solving spatial problems

