

# Analysis, Buffers and Map Algebra [2020]

GEOG 176B: Technical Issues in GIS / Spatial Data

Krzysztof Janowicz

# What You Will Learn in This Lecture

- Understand **buffers** over raster data and different vector primitives.
- Understand design decisions when selecting buffers, e.g., when to **nest buffers** or **dissolve their boundaries**.
- Learn about examples of **proximity analysis**.
- Review terms such as the **scope of operations** or moving windows.
- Understand the basics of **map algebra** and how to apply it to several use cases.

## Some Common Types of Overlays (Recap)

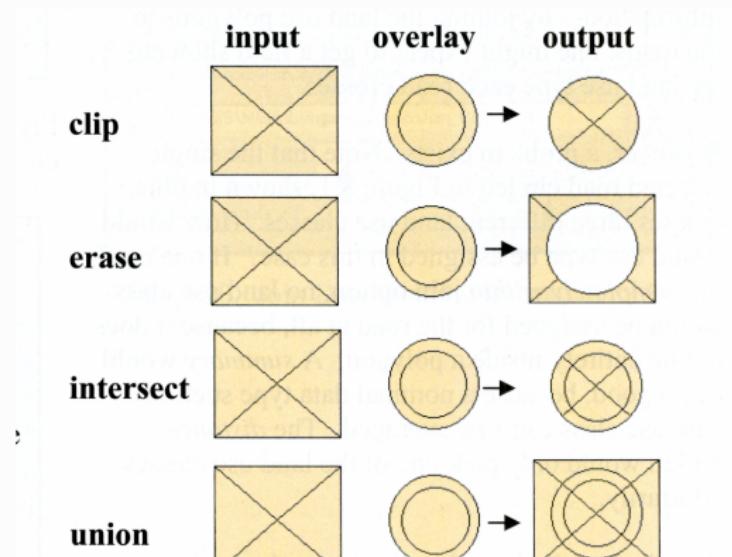
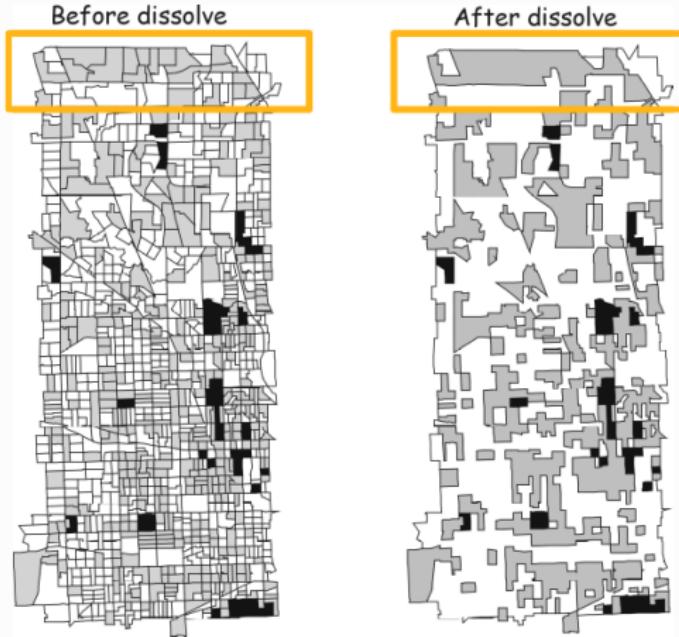


Fig. 8.4. Polygon overlay operations and results. Union and intersect join the attributes of the two input layers. Clip and erase do not.

# Dissolve Operation (Recap)



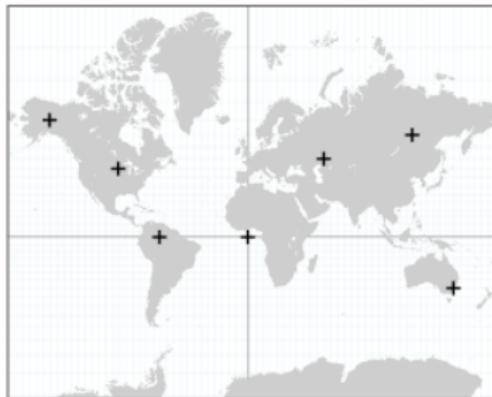
- Creation of **aggregated polygons** based on **merging adjacent polygons** that **share a common attribute value**
- In this neighborhood example, adjacent polygons are merged when they belong to the **same class**.

## Buffers in GIS

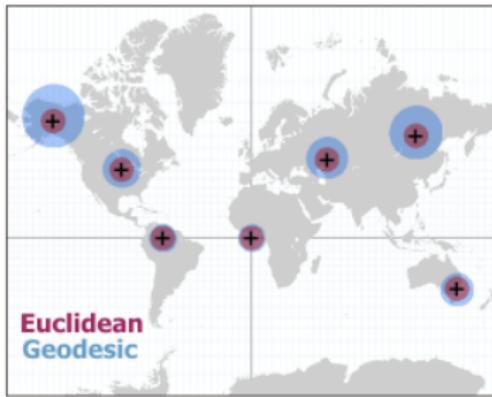
- A **buffer** is a **feature or zone** surrounding a input feature (or zone) that is generated using some **distance** (and distance measure).
- This distance can also be measured in **time** (or other costs).
- Buffers around **vector** geometries, e.g., points and polygons, are very common, but buffers can also be computed for **raster** datasets.
- **Buffers are areas.**
- Buffers can be **combined** when their borders are adjacent or when the buffers overlap.
- Buffers can be **nested** based on different distance bins.
- A common example for the usage of buffers is excluding zones, e.g., around a nature preserve.

# Buffers Distance Methods

World - Mercator

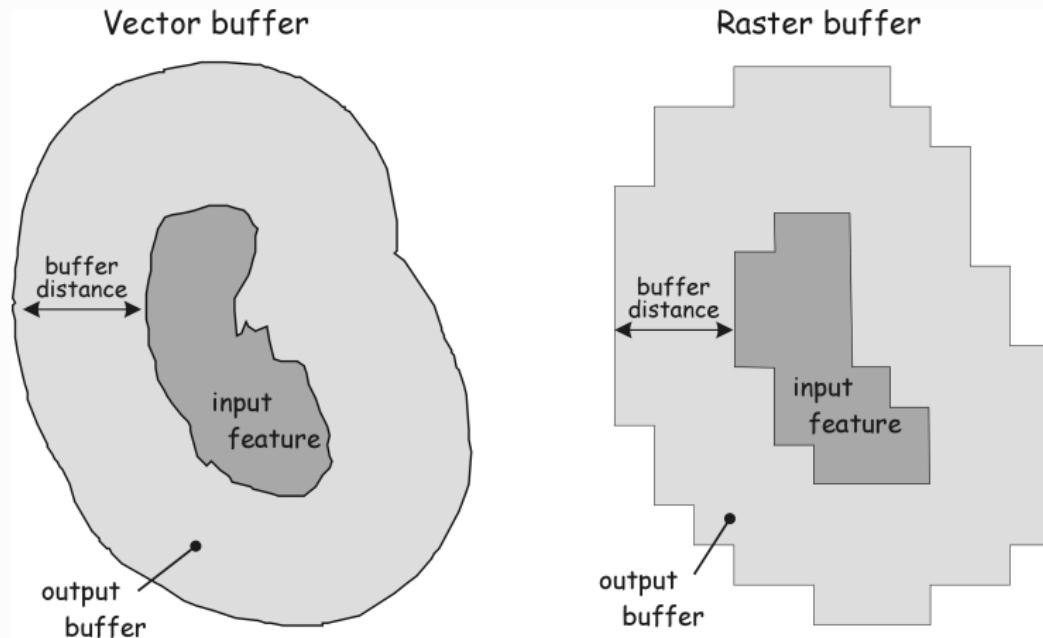


1,000 km Buffer



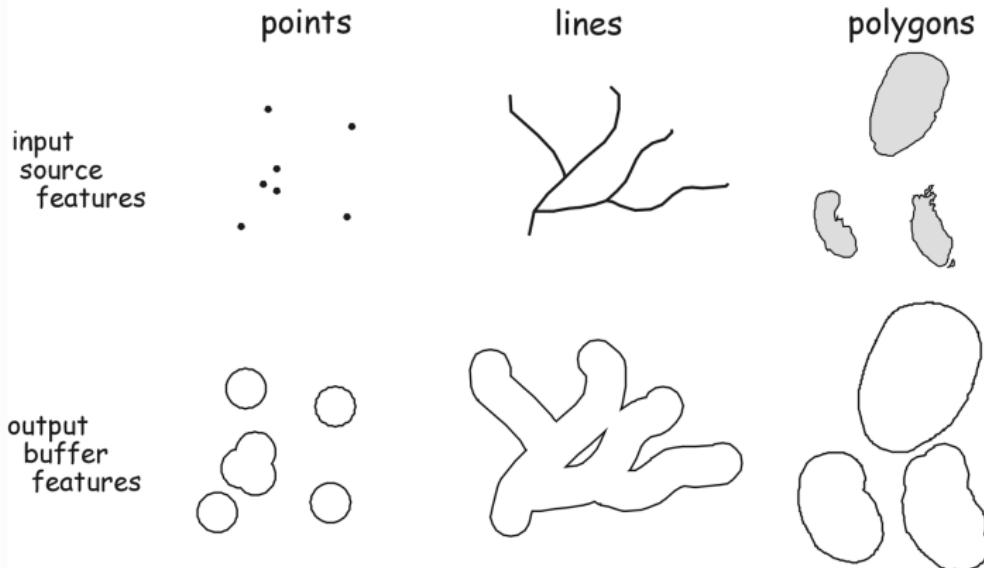
The **Geodesic buffers** (using a geographic coordinate system) the **Euclidean buffers** (using a projected coordinate system) are computed around the same cities; notice how they differ.

# Vector and Raster Buffers



# Vector Buffers

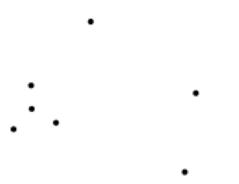
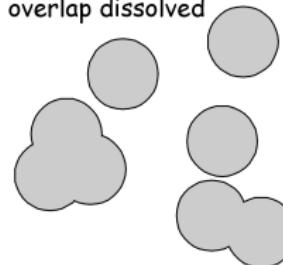
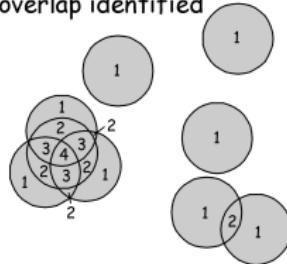
## Vector buffers



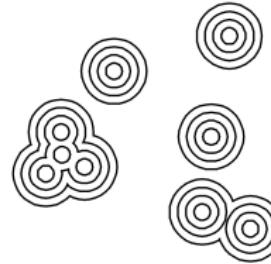
Note how all vector buffers are **polygons**.

# Point Buffers

a) point layer

b) simple buffer,  
overlap dissolvedc) compound buffer,  
overlap identified

d) nested buffers



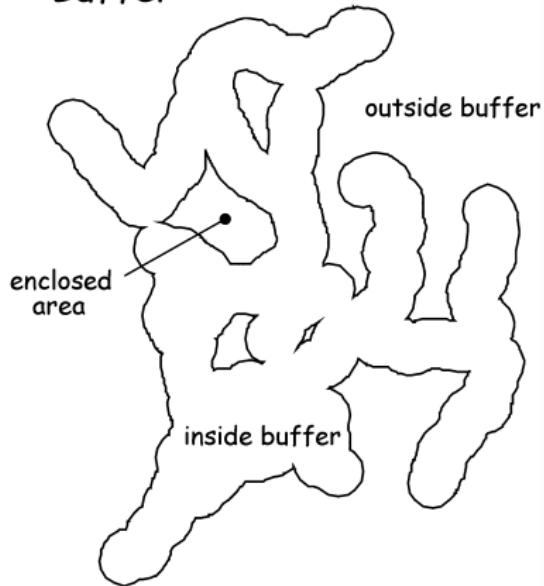
Note how c) translates to densities.

# Line Buffers

Line features

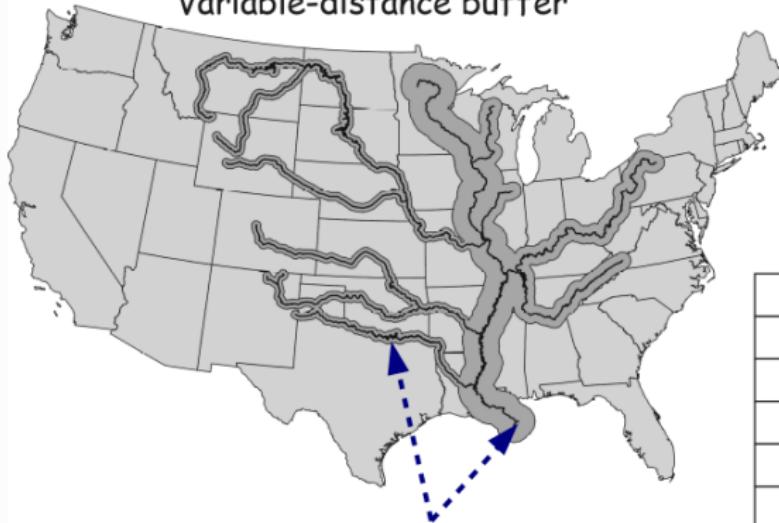


Buffer



# Variable-Distance Buffers

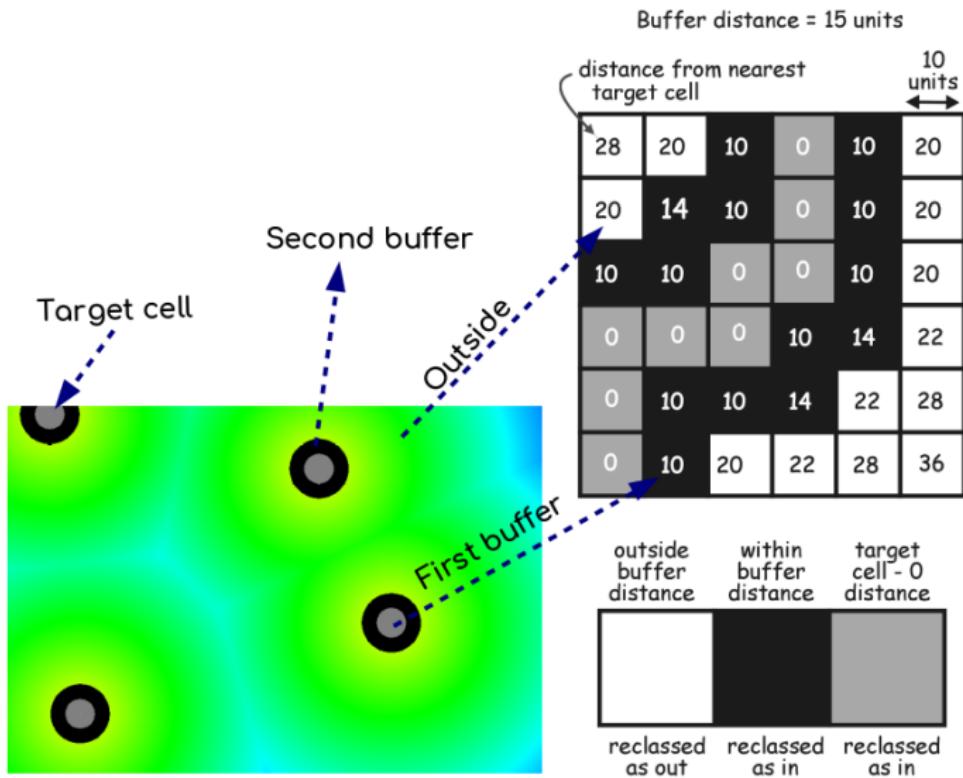
Variable-distance buffer



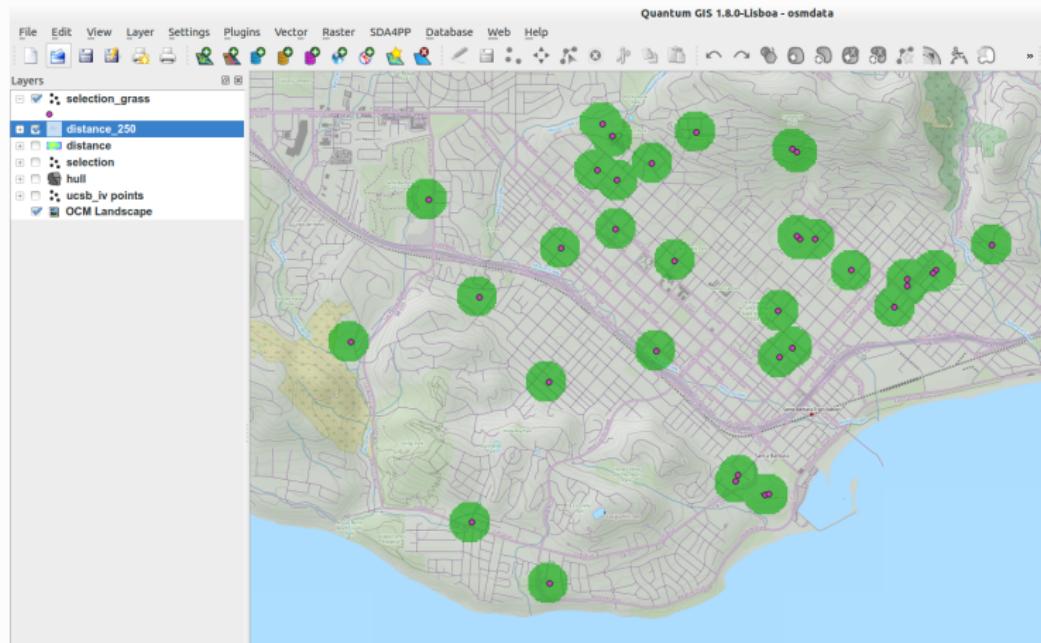
Variable buffer size based on river size

river_identifier	buffdist
mississippi	100
missouri	50
arkansas	50
ohio	75
tennessee	75
st. croix	75
illinois	75
wisconsin	75

# Raster Buffers

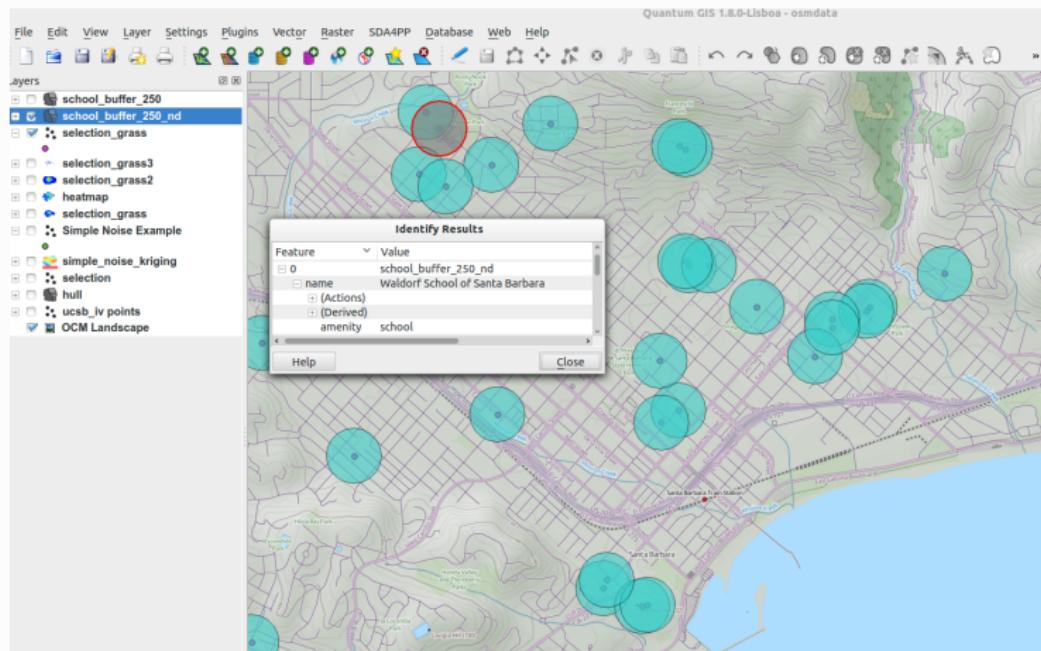


# Raster Buffer Example



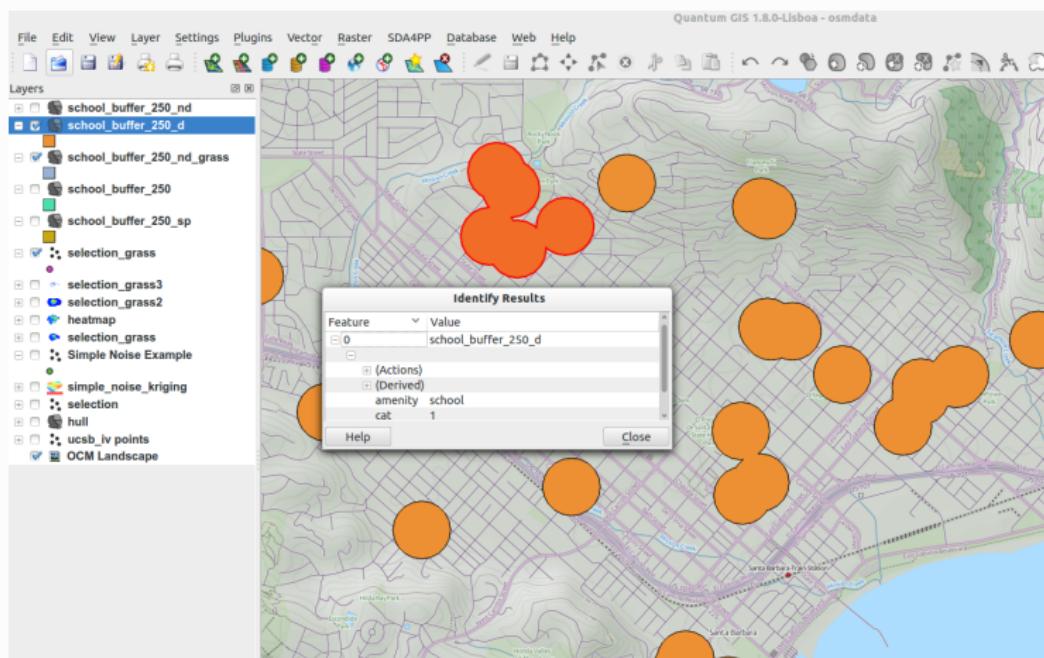
What can be asked of such raster buffer layer; when to use them?

# Point-Based Vector Buffer Example



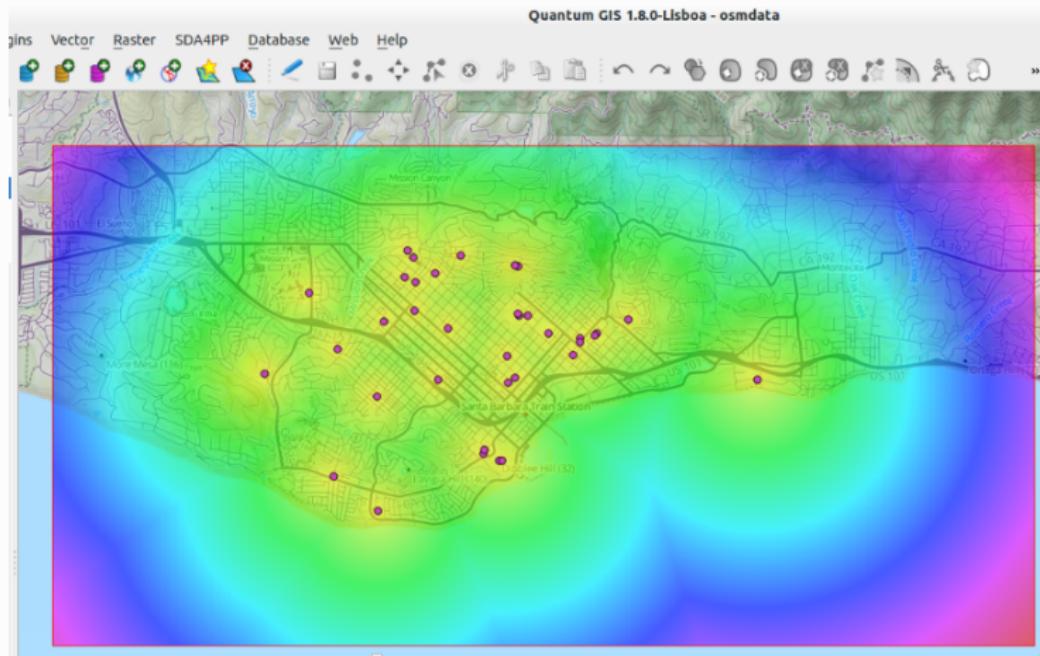
Note how this buffer is about one specific school.

# Point-Based Vector Buffer Example



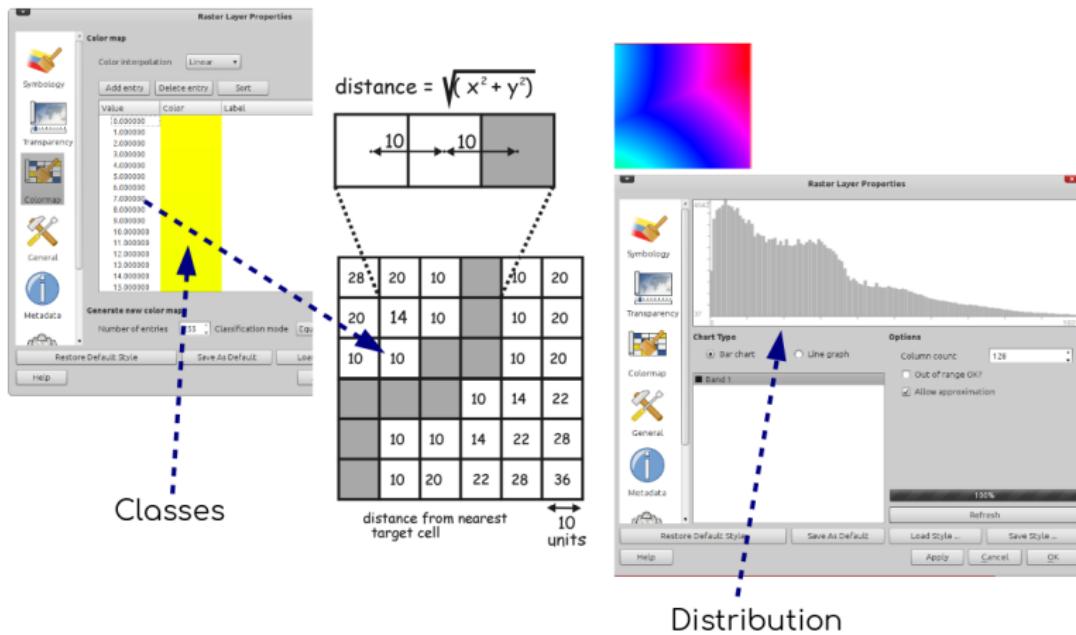
These buffers are **dissolved** using **common** feature type.

# Proximity Map



What do the **colors** mean; what are the **values** of these cells?

# Proximity Map



# Map Algebra

- **Map algebra** is a set-based algebra introduced by Tomlin in the 1980s. It describes how to perform (and combine) primitive operations over **1..\*** geographic (raster) data **layers**
- These operations often include:
  - **Arithmetic operations** such as addition or multiplication
  - **Statistical operations** such as means, maxima or minima
  - **Relational operations** such as greater than or (not) equal
  - ...
- can be used to combine expressions or perform different operations based on some conditional statement (**IF...THEN...**)
- Use cases range from everything starting from data cleaning to visualization.

# Map Algebra – Combining Cell Values

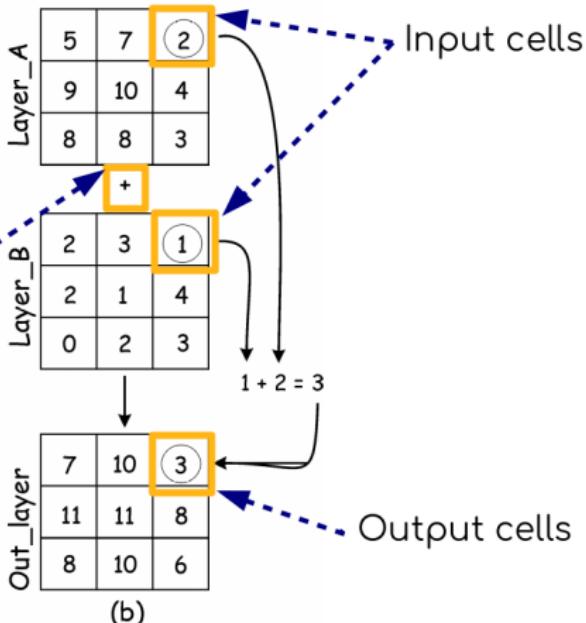
2 x

In_layer	5	7	2	3	(3)
	9	10	4	6	7
	8	8	3	4	3
	7	7	4	4	3
	8	7	4	3	2

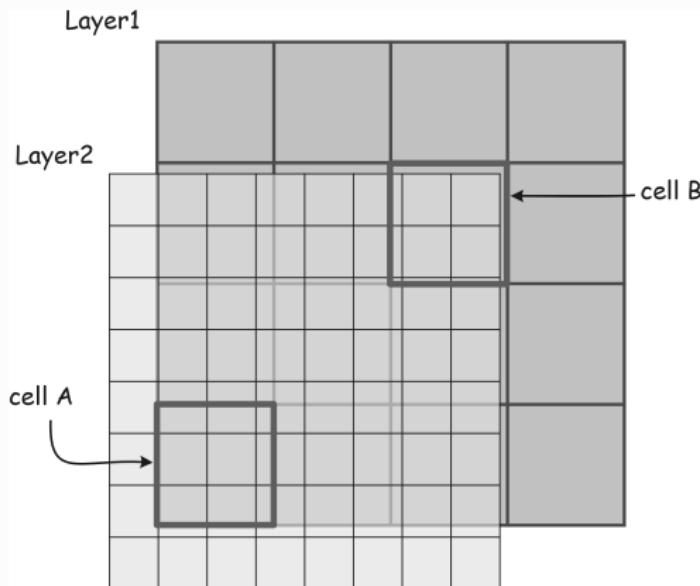
Operation

Out_layer	10	14	4	6	(6)
	18	20	8	12	14
	16	16	6	8	6
	14	14	8	8	6
	16	14	8	6	4

(a)



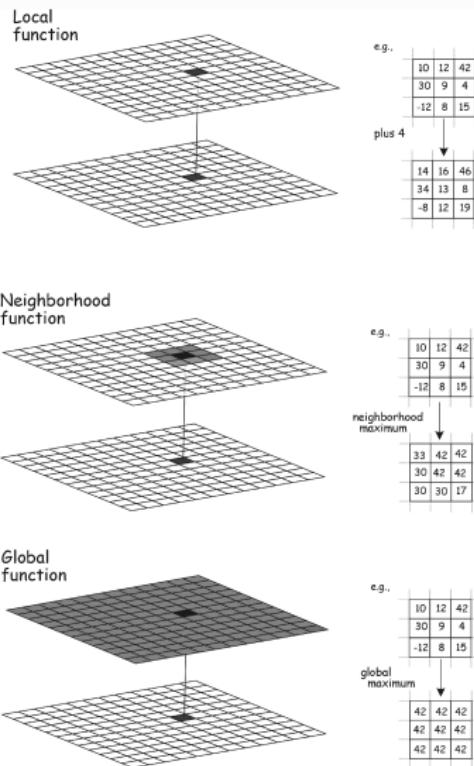
# Map Algebra – Requirements



- Before combining cell values, both layers must be **geo-registered** and of the same spatial **resolution**.
- You may have to **resample** the layer.

# Scopes of Raster Operations

- Similarly, to **vector** operations **raster** operations have a local, neighborhood, or global **scope**.
- **Local** and **neighborhood** scopes are most often used in raster analysis.



# Local Functions – Boolean Operators

Input

a) AND

1	3	1	1
0	N	2	-1
1	2	5	0
0	1	N	N

0	1	0	9
0	5	2	5
0	2	N	2
0	-3	4	8

= ?

b) OR

1	3	1	1
0	N	2	-1
1	2	5	0
0	1	N	N

0	1	0	9
0	5	2	5
0	2	N	2
0	-3	4	8

= ?

c) NOT

1	3	1	1
0	N	2	-1
1	2	5	0
0	1	N	N

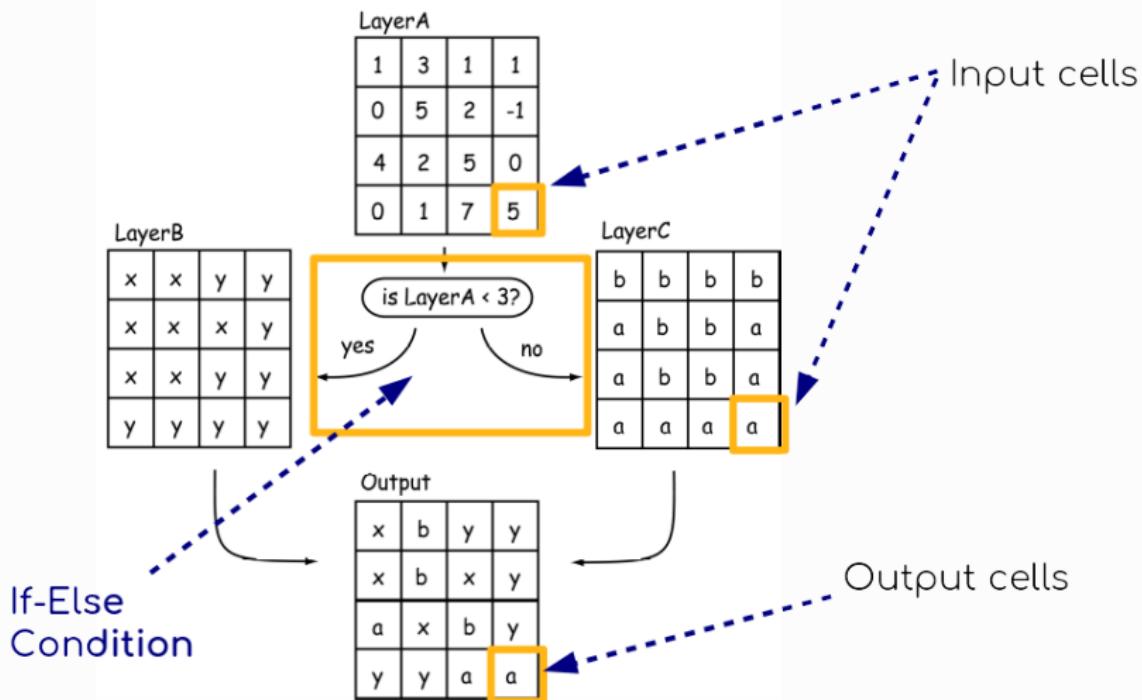
= ?

# Local Functions – Boolean Operators

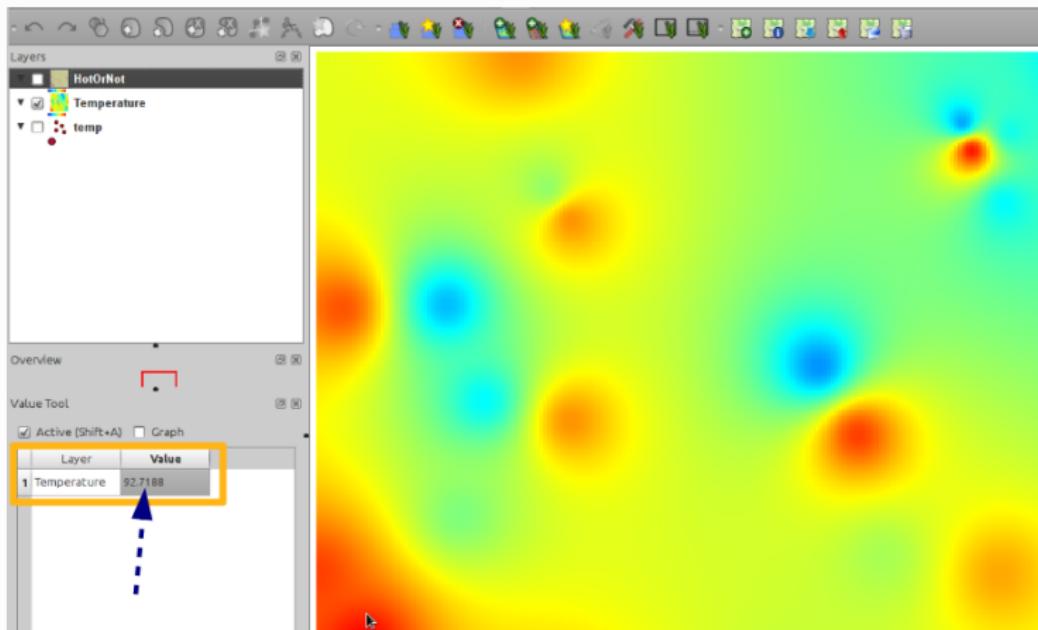
	Input		Output																																																
a)	<table border="1"><tr><td>1</td><td>3</td><td>1</td><td>1</td></tr><tr><td>0</td><td>N</td><td>2</td><td>-1</td></tr><tr><td>1</td><td>2</td><td>5</td><td>0</td></tr><tr><td>0</td><td>1</td><td>N</td><td>N</td></tr></table>	1	3	1	1	0	N	2	-1	1	2	5	0	0	1	N	N	AND	<table border="1"><tr><td>0</td><td>1</td><td>0</td><td>9</td></tr><tr><td>0</td><td>5</td><td>2</td><td>5</td></tr><tr><td>0</td><td>2</td><td>N</td><td>2</td></tr><tr><td>0</td><td>-3</td><td>4</td><td>8</td></tr></table> $=$ <table border="1"><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>N</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>N</td><td>0</td></tr><tr><td>0</td><td>1</td><td>N</td><td>N</td></tr></table>	0	1	0	9	0	5	2	5	0	2	N	2	0	-3	4	8	0	1	0	1	0	N	1	1	0	1	N	0	0	1	N	N
1	3	1	1																																																
0	N	2	-1																																																
1	2	5	0																																																
0	1	N	N																																																
0	1	0	9																																																
0	5	2	5																																																
0	2	N	2																																																
0	-3	4	8																																																
0	1	0	1																																																
0	N	1	1																																																
0	1	N	0																																																
0	1	N	N																																																
b)	<table border="1"><tr><td>1</td><td>3</td><td>1</td><td>1</td></tr><tr><td>0</td><td>N</td><td>2</td><td>-1</td></tr><tr><td>1</td><td>2</td><td>5</td><td>0</td></tr><tr><td>0</td><td>1</td><td>N</td><td>N</td></tr></table>	1	3	1	1	0	N	2	-1	1	2	5	0	0	1	N	N	OR	<table border="1"><tr><td>0</td><td>1</td><td>0</td><td>9</td></tr><tr><td>0</td><td>5</td><td>2</td><td>5</td></tr><tr><td>0</td><td>2</td><td>N</td><td>2</td></tr><tr><td>0</td><td>-3</td><td>4</td><td>8</td></tr></table> $=$ <table border="1"><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>0</td><td>N</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>N</td><td>1</td></tr><tr><td>0</td><td>1</td><td>N</td><td>N</td></tr></table>	0	1	0	9	0	5	2	5	0	2	N	2	0	-3	4	8	1	1	1	1	0	N	1	1	1	1	N	1	0	1	N	N
1	3	1	1																																																
0	N	2	-1																																																
1	2	5	0																																																
0	1	N	N																																																
0	1	0	9																																																
0	5	2	5																																																
0	2	N	2																																																
0	-3	4	8																																																
1	1	1	1																																																
0	N	1	1																																																
1	1	N	1																																																
0	1	N	N																																																
c)	NOT	<table border="1"><tr><td>1</td><td>3</td><td>1</td><td>1</td></tr><tr><td>0</td><td>N</td><td>2</td><td>-1</td></tr><tr><td>1</td><td>2</td><td>5</td><td>0</td></tr><tr><td>0</td><td>1</td><td>N</td><td>N</td></tr></table>	1	3	1	1	0	N	2	-1	1	2	5	0	0	1	N	N	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>N</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>N</td><td>N</td></tr></table>	0	0	0	0	1	N	0	0	0	0	0	1	1	0	N	N																
1	3	1	1																																																
0	N	2	-1																																																
1	2	5	0																																																
0	1	N	N																																																
0	0	0	0																																																
1	N	0	0																																																
0	0	0	1																																																
1	0	N	N																																																

# Introducing Conditions/ Control Statements

Output = CON (LayerA < 3, LayerB, LayerC)

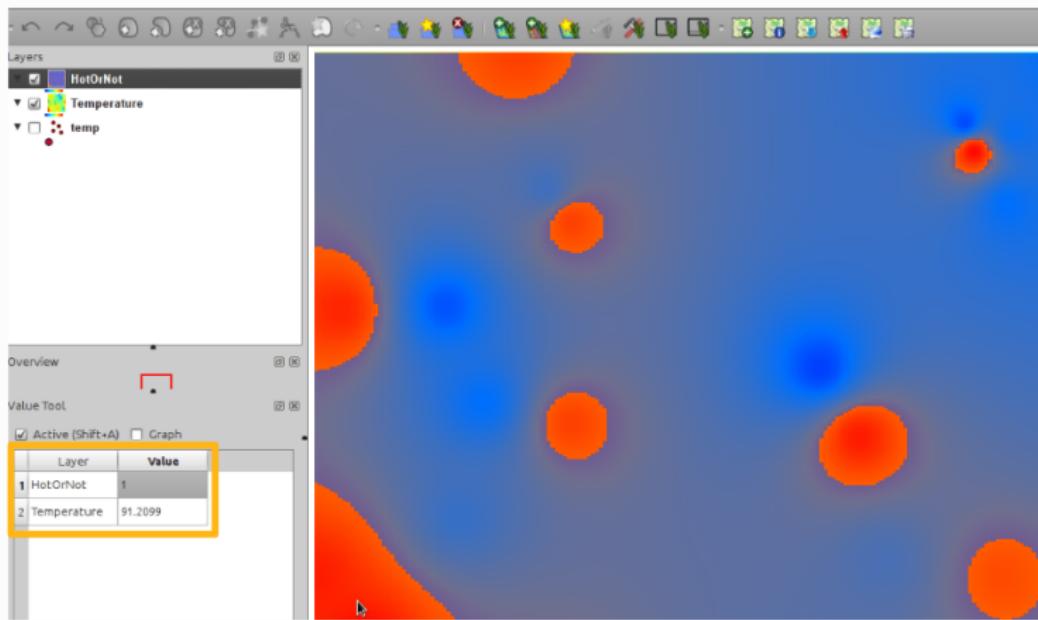


# Classify Temperature Example



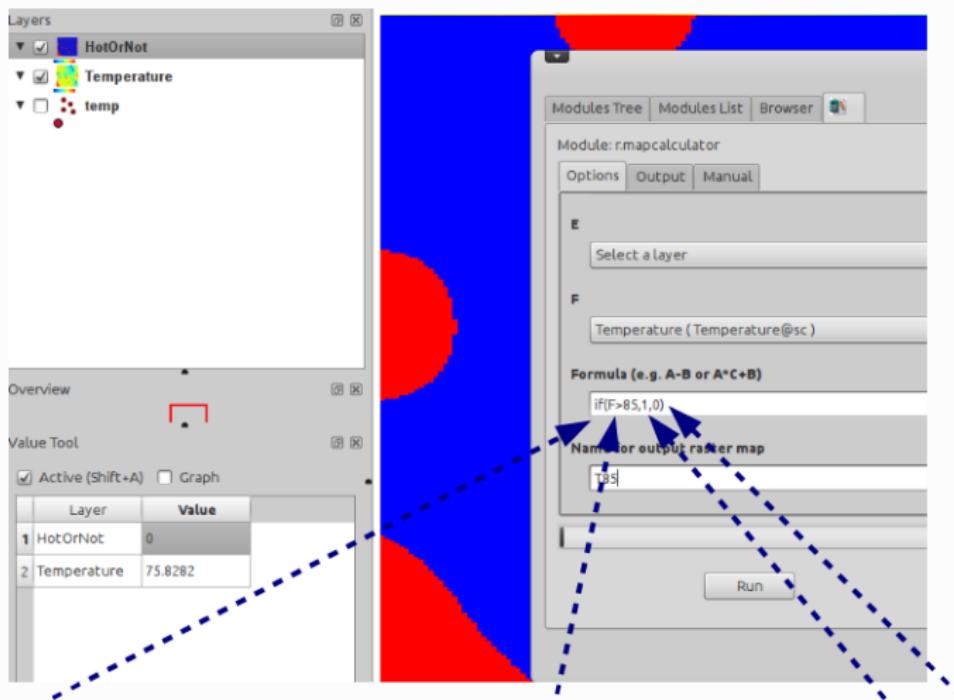
Based on your personal ranking, is this a very warm/hot temperature spot?

# Classify Temperature Example



How to create a raster map showing hot areas based on your ranking?

# Classify Temperature Example



If a cell temperature value is greater than 85, set the cell value to 1, else 0.

## Reclassification Operations

**Input**

1	3	1	1
0	N	2	-1
1	2	5	0
0	1	N	N

Reclass  
by  
table

in	out
0	a
1	x
2	b
3	f
4	c
5	s

**Output**

x	f	x	x
a	N	b	N
x	b	s	a
a	x	N	N

=

1	3	1	1
0	N	2	-1
1	2	5	0
0	1	N	N

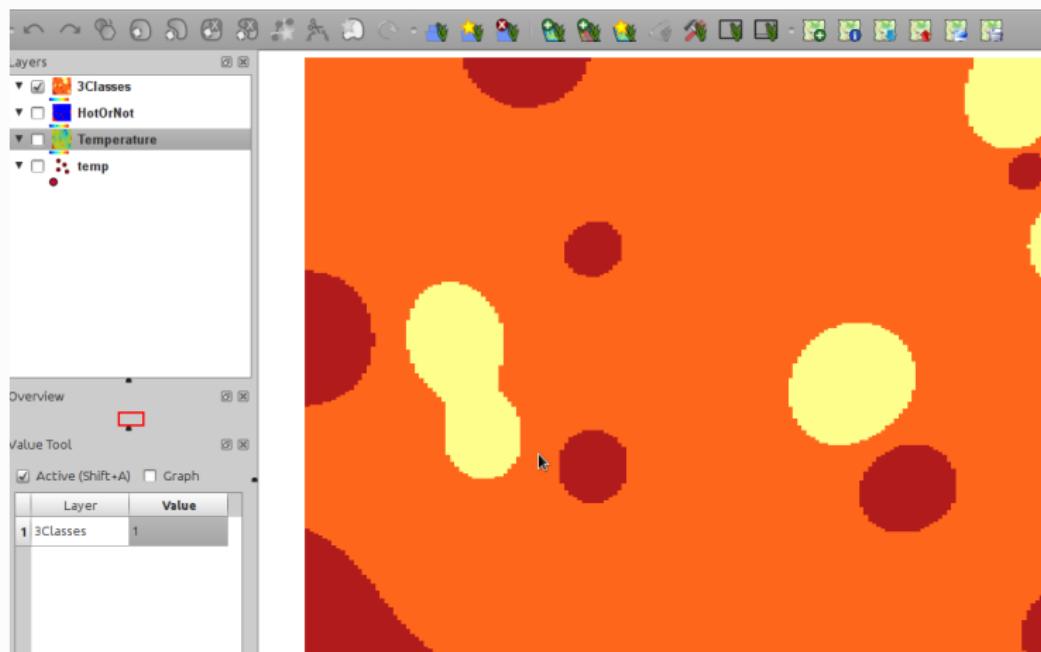
Reclass  
by  
ranges

in range	out
0 to 1.5	a
1.5 to 3.5	b
3.5 to 10	c
N	d

=

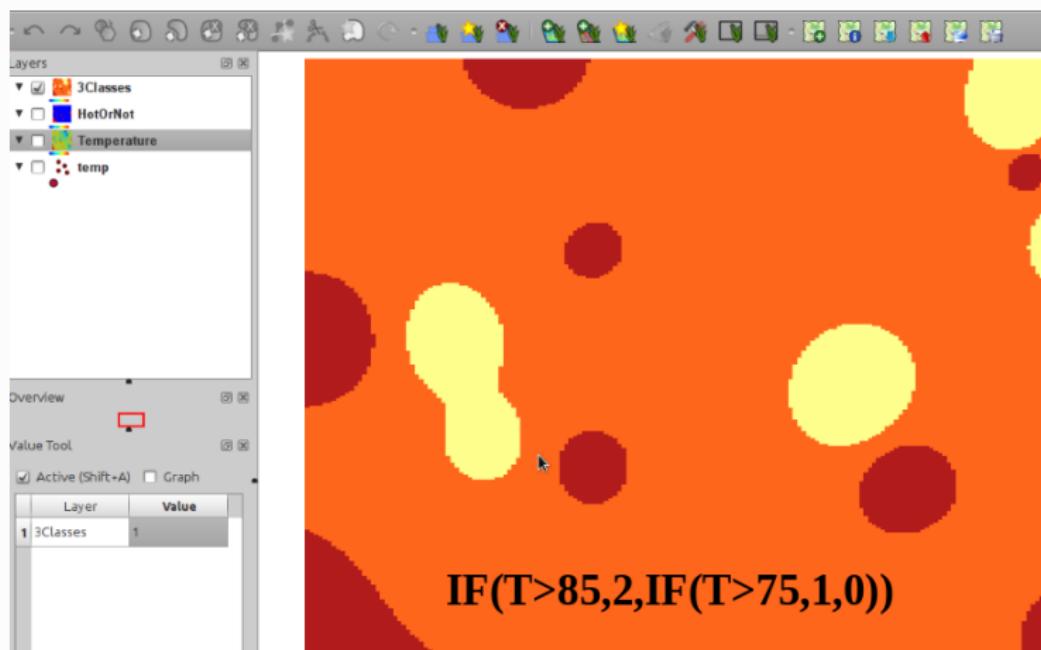
a	b	a	a
a	d	b	N
a	b	c	a
a	b	d	d

## Classify Temperature Example with 3 Classes



How to create a raster map showing **hot, warm, and cold** areas ?

## Classify Temperature Example with 3 Classes



How to create a raster map showing **hot, warm, and cold** areas ?

# The Map Algebra Realization of a Clip Overlay

Input raster

2	2	2	8	8	2	2	2
2	2	2	8	8	8	2	2
2	3	3	3	8	8	8	7
2	3	3	3	8	8	8	7
3	3	3	6	6	6	7	7
3	3	3	3	6	6	6	7
3	6	3	6	6	6	6	6
3	6	6	6	6	6	6	6

x

Clip raster

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	1	1	1	1	1	0
0	0	1	1	1	1	0	0
0	0	1	1	1	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

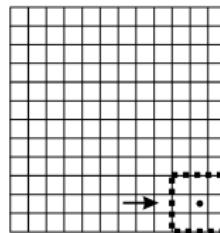
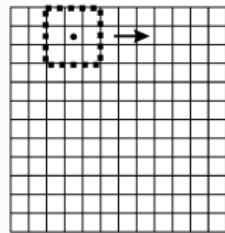
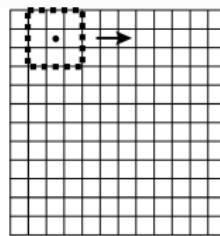
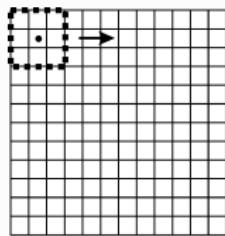


Output raster

0	0	0	0	8	2	2	2
0	0	0	0	8	8	2	2
0	0	3	3	8	8	8	0
0	0	3	3	3	8	0	0
0	0	3	6	6	0	0	0
0	0	3	3	0	0	0	0
0	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0

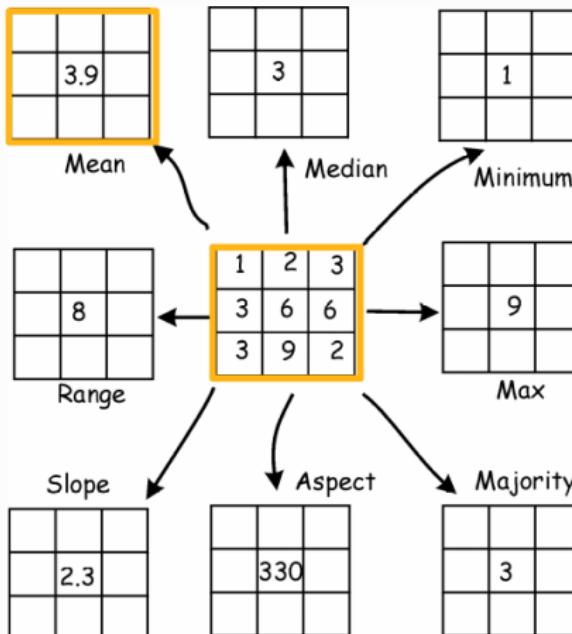
How to use just one operator to realize an **erase overlay**?

# Neighborhood Functions – Moving Window



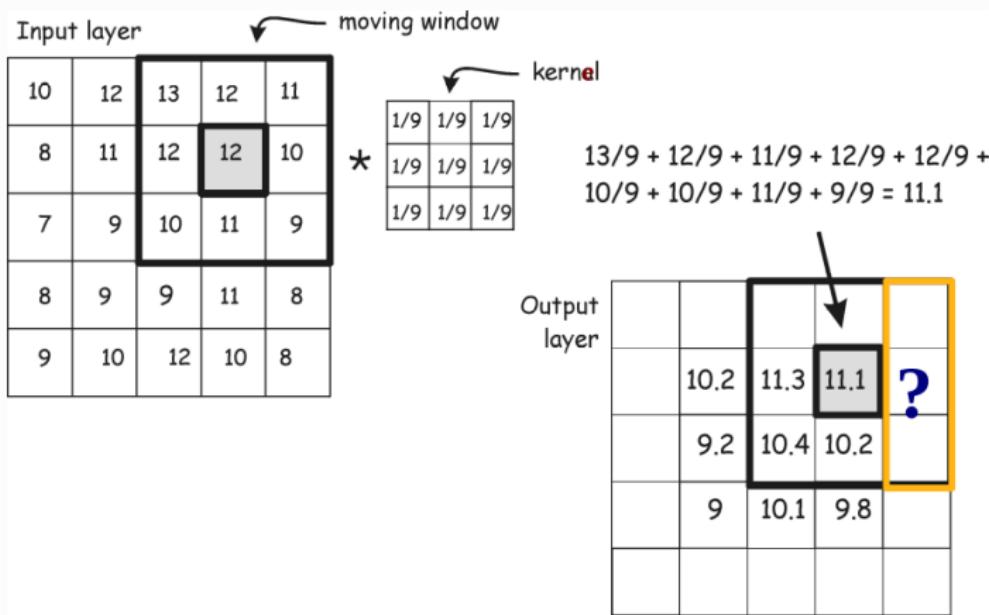
- The **value** for the cell in the **center** of each window is **computed** by all the **3x3 neighbor cells**.
- The **window moves** cell by cell for each row and column
- **Why 3x3 or 5x5 cells but not 4x4 cells?**

# Moving Windows and Neighborhood Functions



How would you realize the **mean** using map algebra?

# Kernels



A **kernel** is the set of **constants** for the cell values in a given **window**.

# Different Kernels for Corners and Margins

## Mean function kernels

corner

$\frac{1}{4}$	$\frac{1}{4}$
$\frac{1}{4}$	$\frac{1}{4}$

margin

1/6	1/6	1/6
1/6	1/6	1/6

corner

$\frac{1}{4}$	$\frac{1}{4}$
$\frac{1}{4}$	$\frac{1}{4}$

margin

1/6	1/6
1/6	1/6
1/6	1/6

main

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

margin

1/6	1/6
1/6	1/6
1/6	1/6

corner

$\frac{1}{4}$	$\frac{1}{4}$
$\frac{1}{4}$	$\frac{1}{4}$

margin

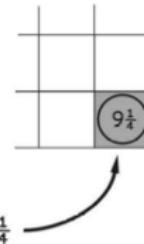
1/6	1/6	1/6
1/6	1/6	1/6

corner

$\frac{1}{4}$	$\frac{1}{4}$
$\frac{1}{4}$	$\frac{1}{4}$

example application,  
lower right corner

10	12	13	12	11
8	11	12	12	10
7	9	10	11	9
8	9	9	11	8
9	10	12	10	8



$$\frac{1}{4} \cdot 11 + \frac{1}{4} \cdot 8 + \frac{1}{4} \cdot 10 + \frac{1}{4} \cdot 8 = 9\frac{1}{4}$$

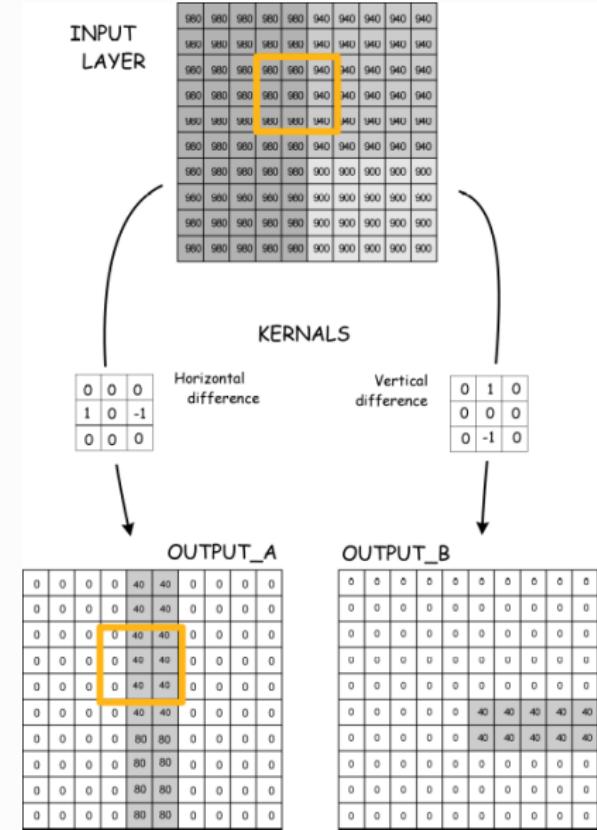
Select a different kernel window for corners or margins or a larger study area.

# Kernel-Based Edge Detection

- Detect **abrupt changes** in cell values using a kernel.

- **Example:**

$$\begin{aligned}(980 * 1) + (940 * -1) \\= 980 - 940 \\= 40\end{aligned}$$



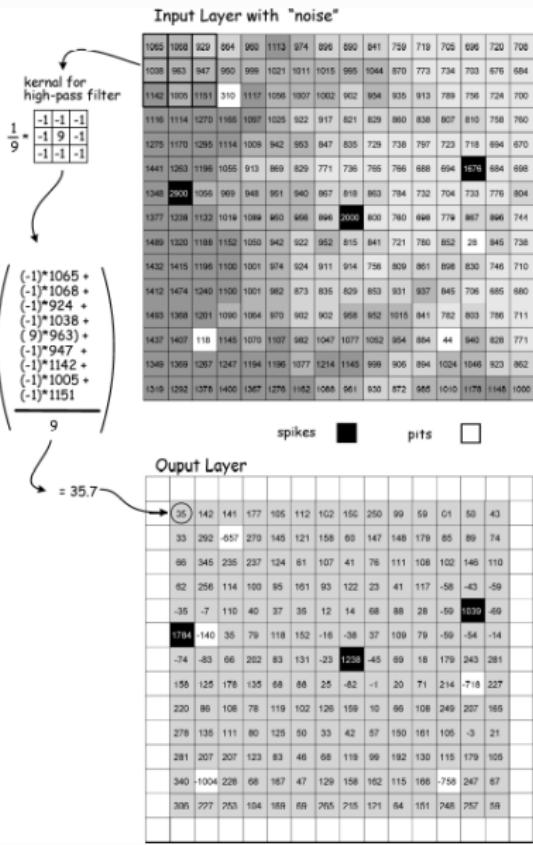
# Kernel-based High-Pass Filter

100	100	100
100	100	100
100	100	100

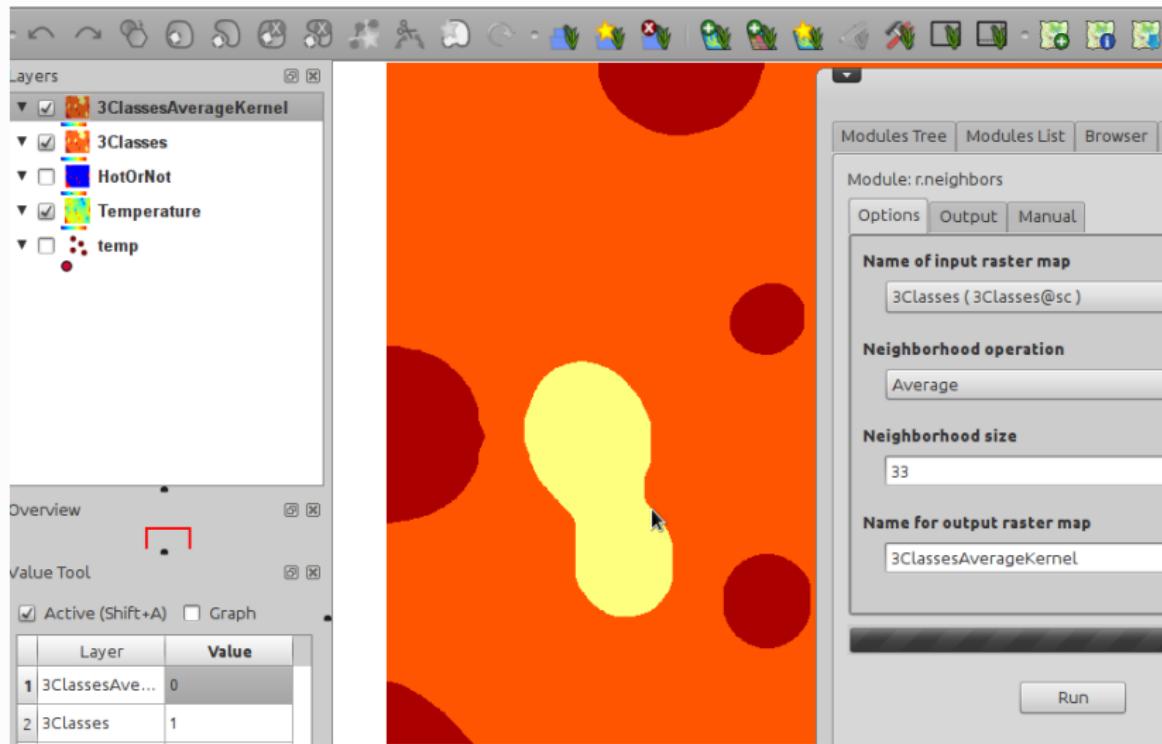
$$(-800+900) / 9 = 11$$

100	100	100
100	1000	100
100	100	100

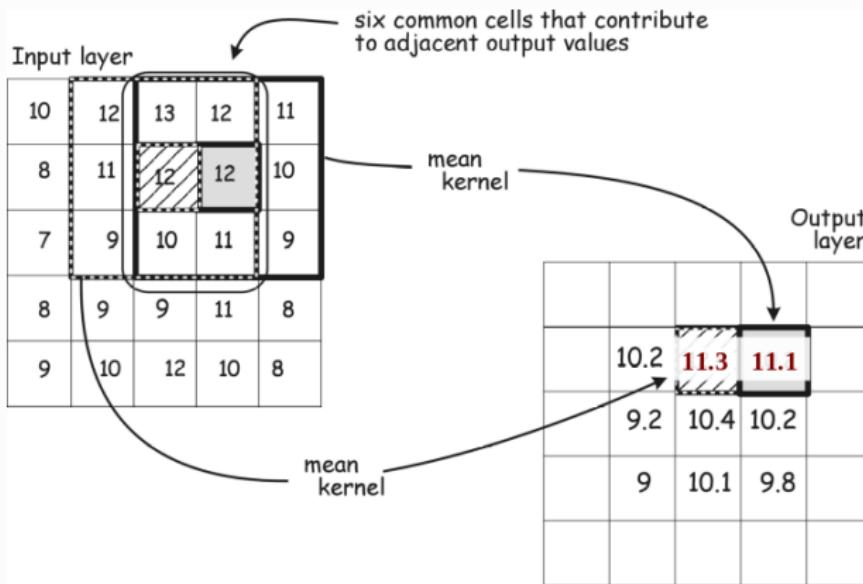
$$(-800+9000) / 9 = 911$$



# Using an Averaging Kernel for Smoothing



# Kernel Functions and Spatial Covariance

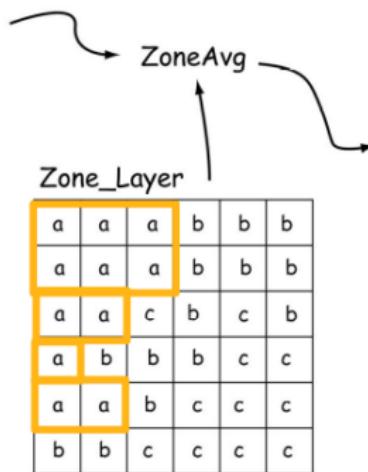


Keep in mind that kernel functions will **increase** the **spatial covariance** of cell values. For adjacent cells, **6 out of 9 cells** used to compute the new value will be the same.

# Zonal Functions

In_Layer	7	4	2	3	6	10
1	3	1	4	1	9	
1	4	11	10	8	7	
3	1	2	7	8	10	
2	2	11	7	10	12	
5	8	10	9	13	11	

Out\_Layer = ZoneAvg (In\_Layer, Zone\_Layer)



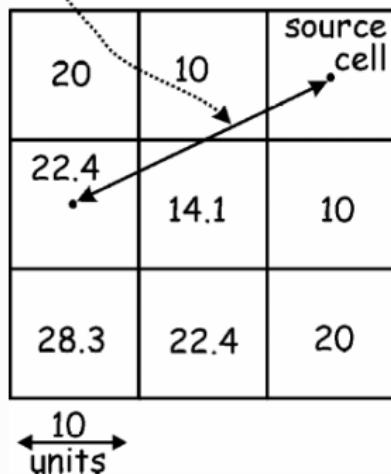
Out_Layer	2.7	2.7	2.7	6.6	6.6	6.6
	2.7	2.7	2.7	6.6	6.6	6.6
	2.7	2.7	9.2	6.6	9.2	6.6
	2.7	6.6	6.6	6.6	9.2	9.2
	2.7	2.7	6.6	9.2	9.2	9.2
	6.6	6.6	9.2	9.2	9.2	9.2

The term **neighborhood** may also be **generalized** for **regions** or zones.

# Cost Surfaces As Analogy to Network Analysis

$$\text{distance} = \sqrt{x^2 + y^2}$$

e.g.,  $D = \sqrt{(20^2 + 10^2)}$   
 $= 22.4$



$$\text{cost} = \text{distance} * \text{fixed cost factor}$$

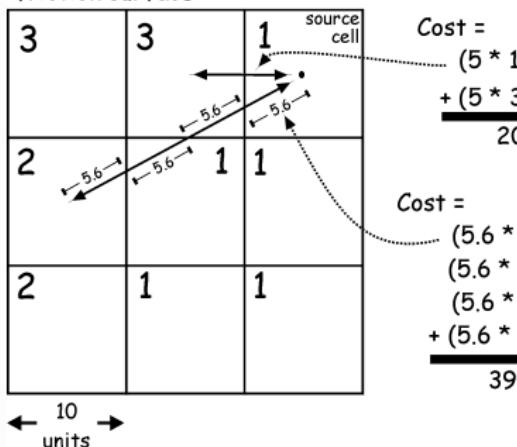
e.g.,  
 $\text{cost} = \text{distance} * 2$

40	20	source cell
44.8	28.2	20
56.6	44.8	40

# Cost Surfaces As Analogy to Network Analysis

cost = cell distance \* friction

friction surface



output cost surface

50	20	source cell
39.1	14.1	10
42.3	22.4	20

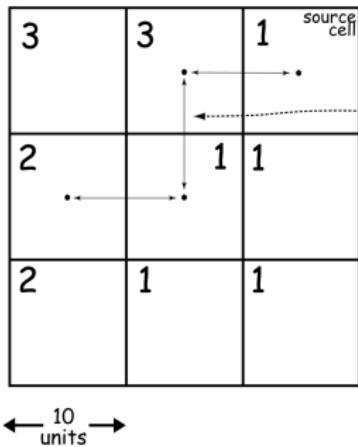
← 10 units →

Instead of a cost constant, **costs** can be modeled as **cell values**.

# Row-Column Distance

$$\text{cost} = \text{row/column distance} * \text{friction}$$

friction surface



Cost =

$$\begin{aligned} & (5 * 1) \\ & + (5 * 3) \\ & + (5 * 3) \\ & + (5 * 1) \\ & + (5 * 1) \\ & + (5 * 2) \end{aligned}$$

---

55

output cost surface

50	20	source cell
55	40	10
45	30	20

← 10 units →

All edges have the same length – either **5 or 10** units.