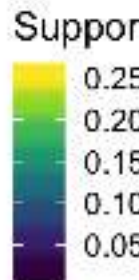
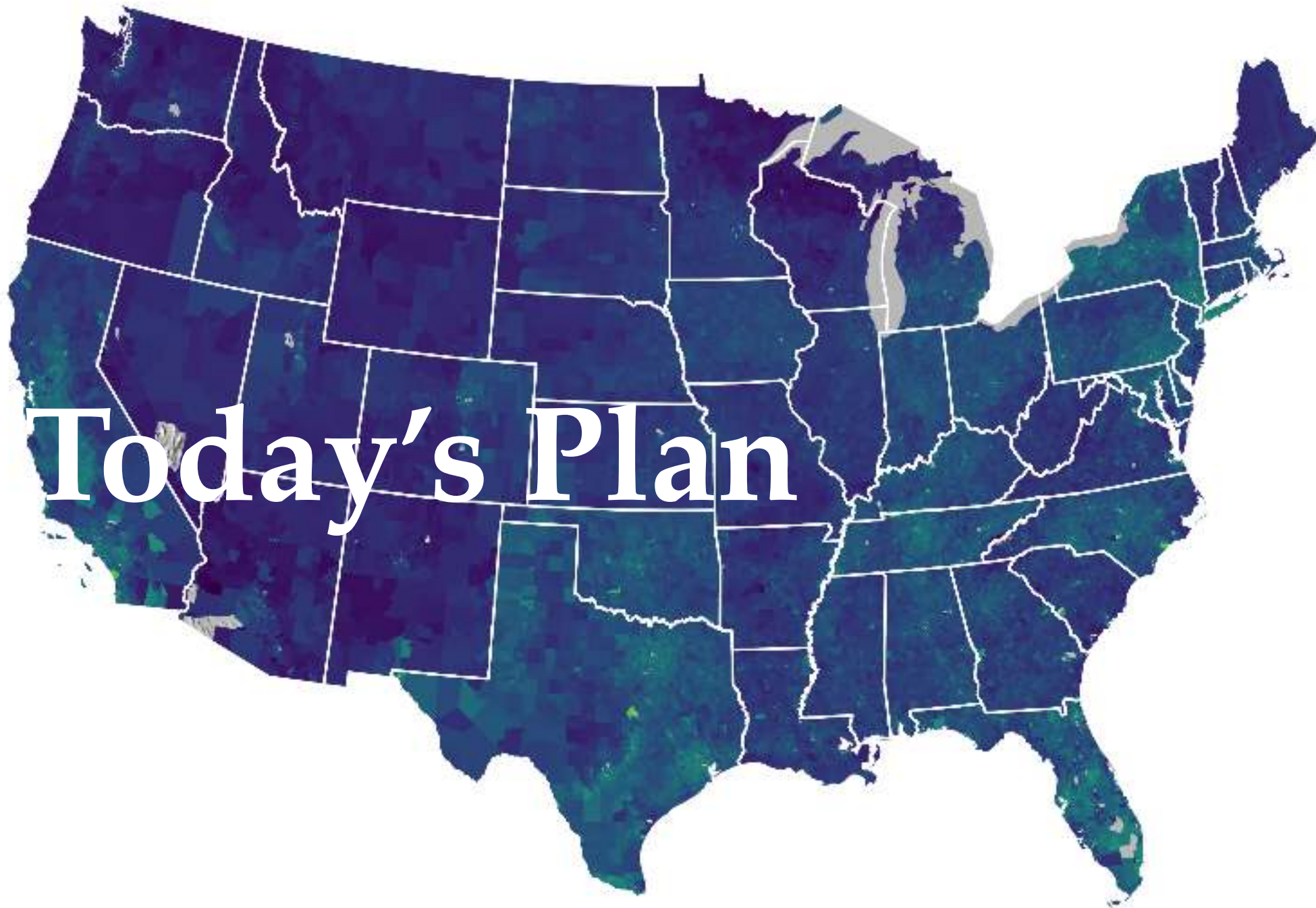


# Raster Data: II

HES 505 Fall 2023: Session 14

Matt Williamson



# Objectives

- By the end of today, you should be able to:
  - Use moving windows as a means of smoothing raster data
  - Reclassify data using conditional statements and reclassification tables
  - Use raster math as a means of creating new data based on an existing dataset.

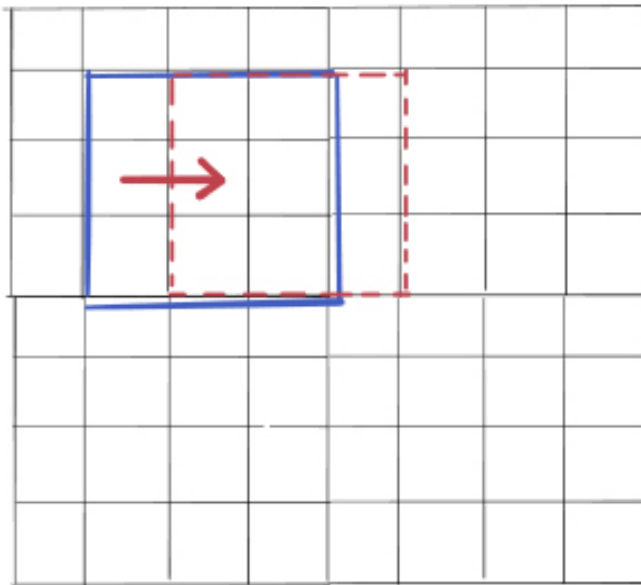
# Moving Windows

# Why use moving windows?

- To create new data that reflects “neighborhood” data
- To smooth out values
- To detect (and fill) holes or edges
- Change the thematic scale of your data (without changing resolution)

# What is a moving window?

## Sliding Window Operations



Inside the **window** the center cell is replaced by the a weighted sum of its neighbors. Calculations are repeated as the **window slides** across the remaining cells.

pygis.io

@bynasa

# Implementing Moving Windows in R

- Use the `focal` function in `terra`

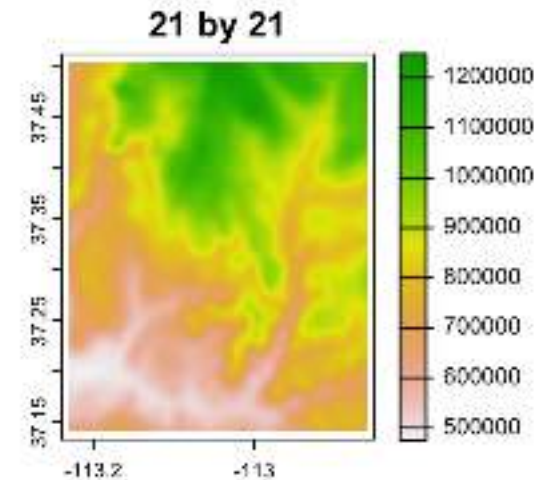
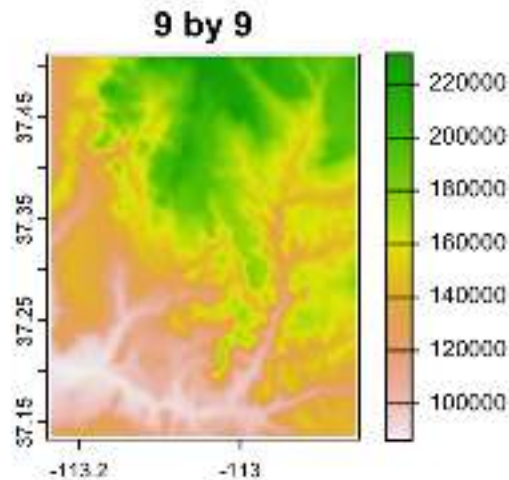
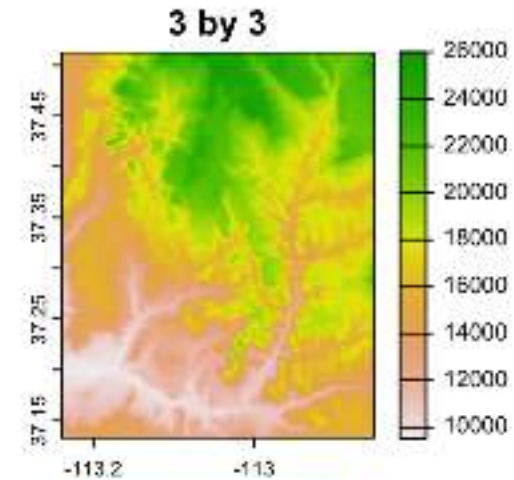
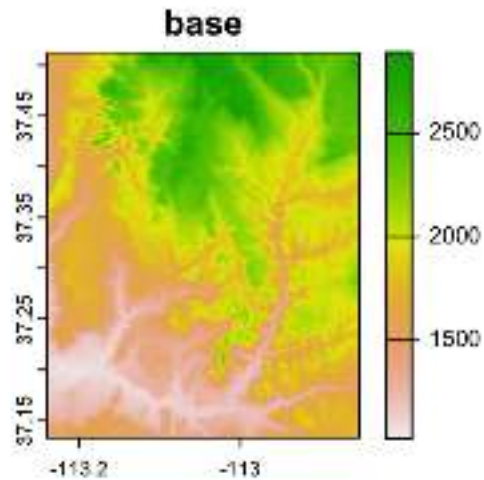
```
focal(x, w=3, fun="sum", ...,  
na.policy="all", fillvalue=NA,  
expand=FALSE, silent=TRUE, filename="",  
overwrite=FALSE, wopt=list())
```

# focal for Continuous Rasters

```
1 library(tidyverse)
2 library(terra)
3 library(spData)
4 srtm = rast(system.file("raster/srtm.tif", package = "spDataLarge"))
5 srtm3  <- focal(x = srtm, w = 3)
6 srtm9  <- focal(x = srtm, w = 9)
7 srtm21 <- focal(x = srtm, w = 21)
```



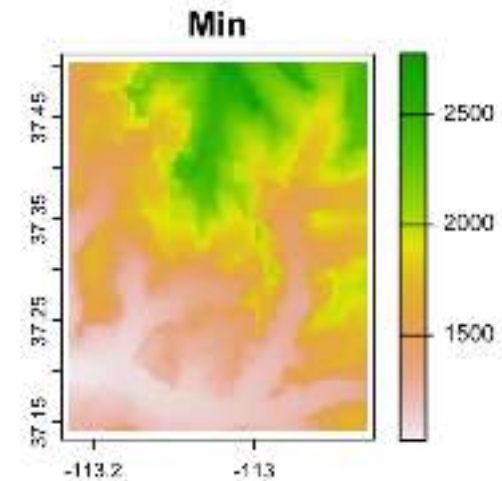
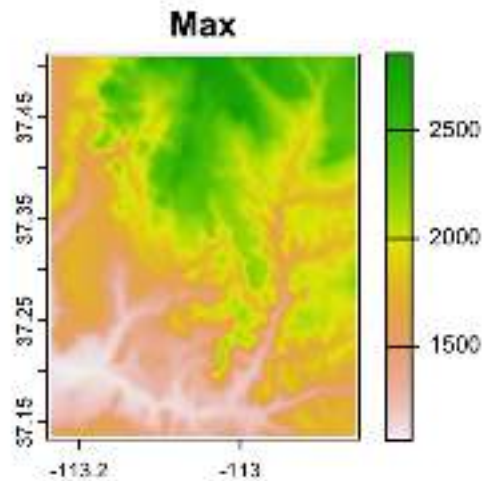
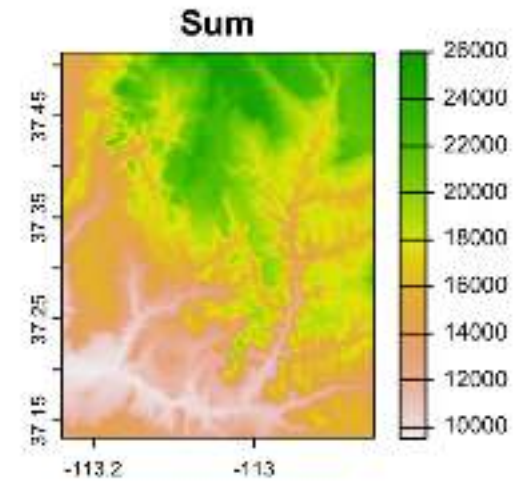
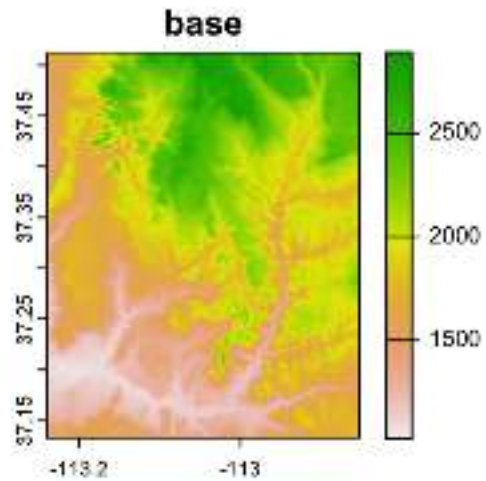
# focal for Continuous Rasters



# focal for Continuous Rasters

```
1 srtmsum  <- focal(x = srtm, w = 3, fun="sum")
2 srtmmax  <- focal(x = srtm, w = 9, fun="mean")
3 srtmmin  <- focal(x = srtm, w = 21, fun="min")
```

# focal for Continuous Rasters



# **focal** for Continuous Rasters

- can alter the size and shape of window by providing a weights matrix for **w**
- Can create different custom functions for **fun** (see the help file)
- **na.policy** for filling holes or avoiding them

# Reclassification

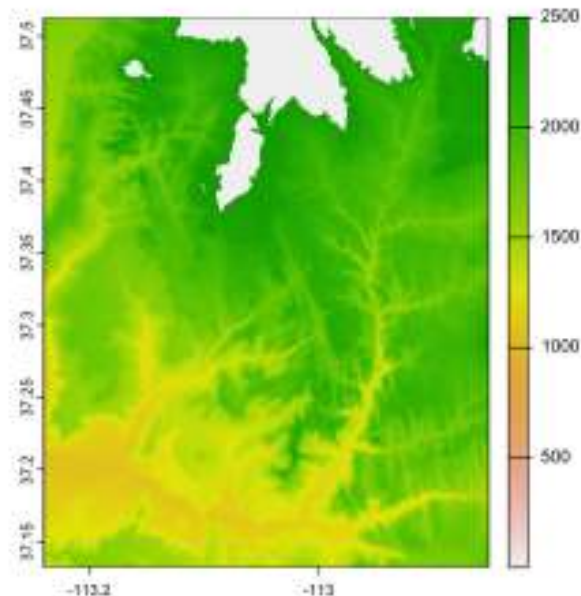
# Reclassification

- Create new data based on the presence of a particular class(es) of interest
- Combine classes in a categorical map
- Useful as inputs for overlay analyses

# Reclassifying rasters in R

- Using `[]` and conditionals

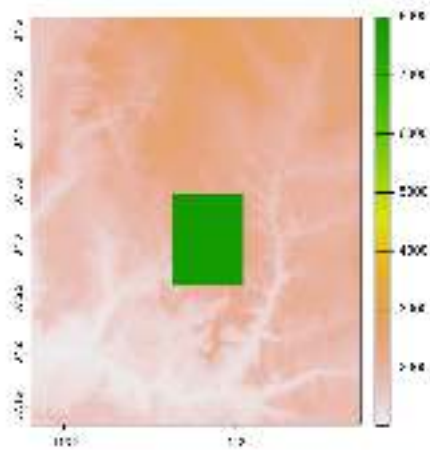
```
1 srtm = rast(system.file("raster/srtm.tif", package = "spDataLarge"))
2 srtm.lowelev <- srtm
3 srtm.lowelev[srtm.lowelev > 2500] <- 1
4 plot(srtm.lowelev)
```



# Reclassifying rasters in R

- Using `[]` and conditionals

```
1 srtm = rast(system.file("raster/srtm.tif", package = "spDataLarge"))
2
3
4 srtm.na <- srtm
5 srtm.na[200:300, 200:300] <- NA
6 srtm.na[is.na(srtm.na)] <- 8000
7 plot(srtm.na)
```



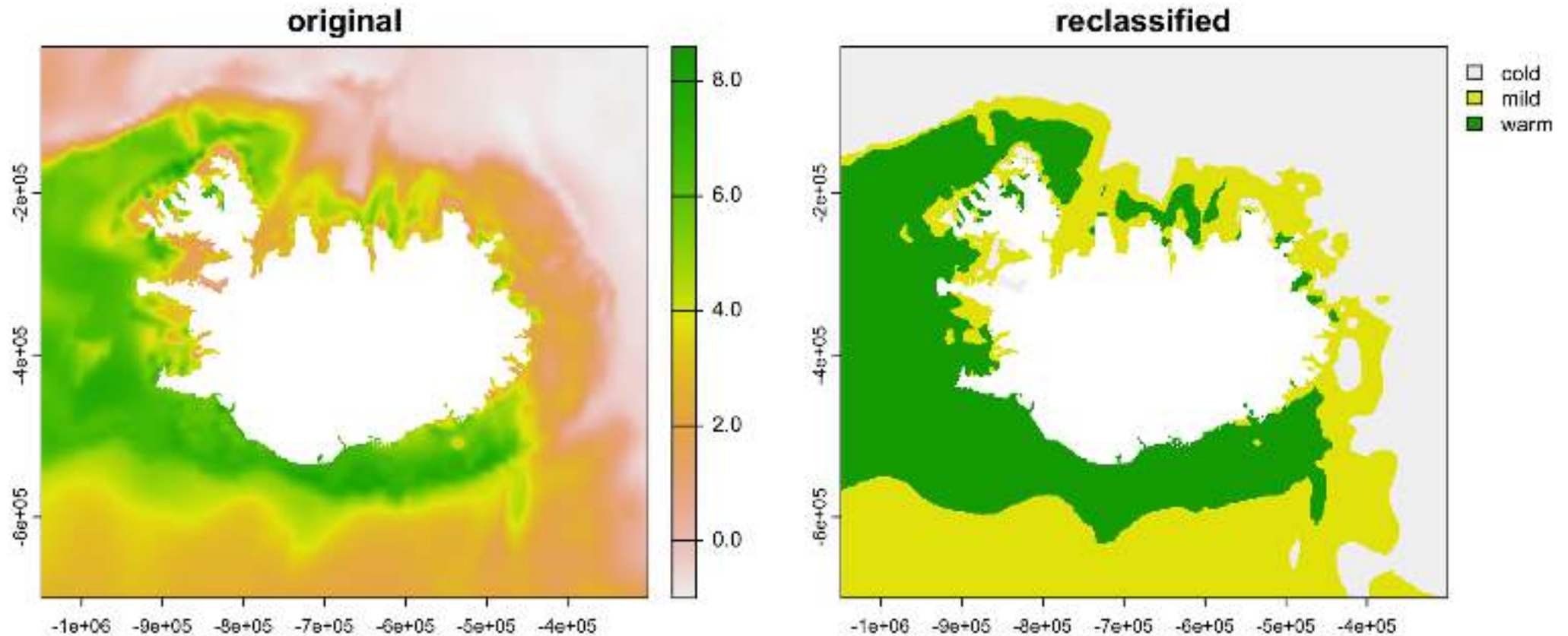


# Reclassifying Categorical Rasters

- Need a classification matrix
- Use `classify`

```
1 mintemp <- rast("ftp://ftp.hafro.is/pub/data/rasters/Iceland_minbtemp.tif")
2 cm <- matrix(c(
3   -2, 2, 0,
4    2, 4, 1,
5    4, 10, 2), ncol = 3, byrow = TRUE)
6
7 # Create a raster with integers
8 temp_reclass <- classify(mintemp, cm)
9 tempcats <- c("cold", "mild", "warm")
10 levels(temp_reclass) <- tempcats
```

# Reclassifying Categorical Rasters

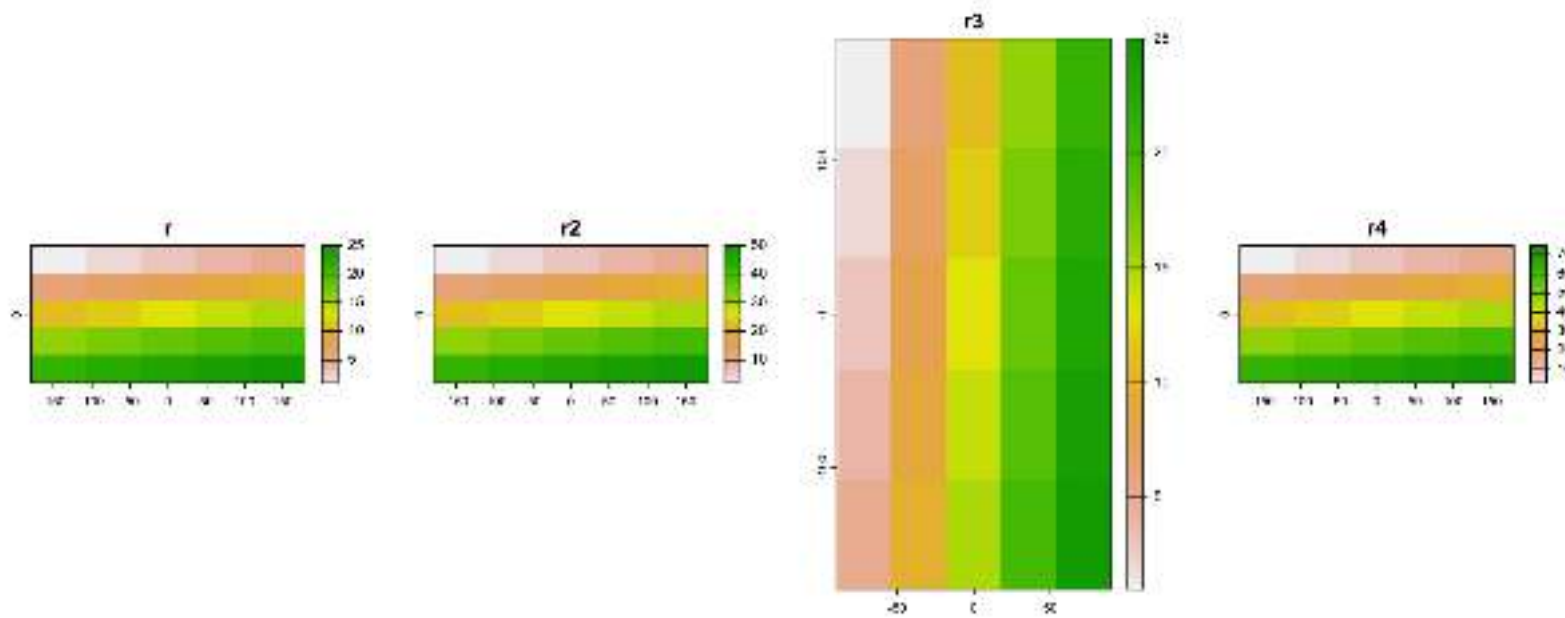


# Raster Math

- Performs cell-wise calculations on 1 (or more) **SpatRasters**
- Generally works the same as matrix operations
- All layers must be aligned

# Raster Math

```
1 r <- rast(ncol=5, nrow=5)
2 values(r) <- 1:ncell(r)
3 r2 <- r*2
4 r3 <- t(r)
5 r4 <- r + r2
```



# Cell-wise operations

- **terra** has a special set of **apply** functions
- **app**, **lapp**, **tapp**
- **app** applies a function to the values of each cell
- **lapp** applies a function using the layer as the value
- **tapp** applies the function to a subset of layers

# Context-specific Functions

- **distance** and relatives are based on relationships between cells
- **terrain** allows calculation of slope, ruggedness, aspect using elevation rasters
- **shade** calculates hillshade based on terrain

