# Operations With Raster Data II

HES 505 Fall 2022: Session 12

Matt Williamson

# Today's Plan

# Objectives

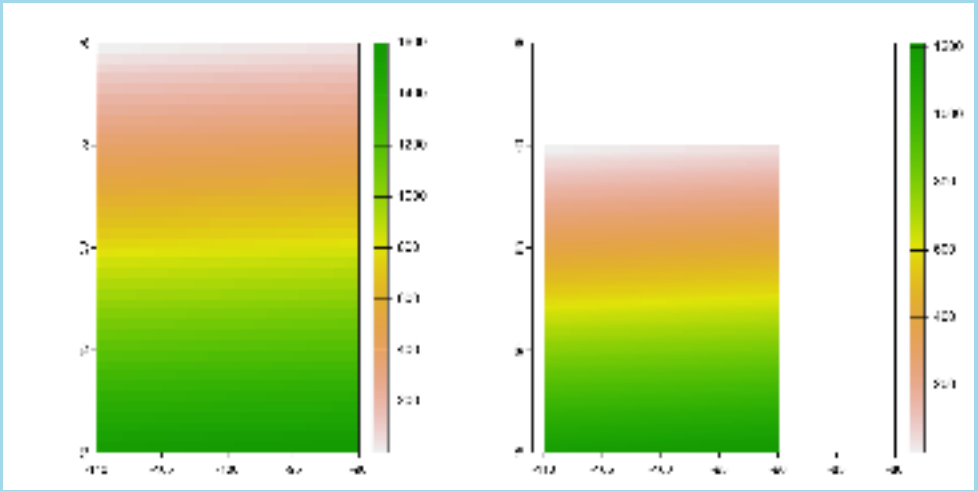
- By the end of today, you should be able to:
  - Access and manipulate cell values
  - Generate new rasters using mathematical functions
  - Summarize rasters using global functions
  - Generate new rasters describing the spatial context of individual cells

# Revisiting Projections

# Revisiting Projections

- `resample` transfers values between SpatRaster objects that do not align
- Must have same `crs`

```
a <- rast(ncols=40, nrows=40, xmin=-110, xmax=-90,
          crs="+proj=longlat +datum=WGS84")
values(a) <- 1:ncell(a)

b <- rast(ncols=94, nrows=124, xmin=-111, xmax=-80
w <- resample(a, b)
```



```
origin(a)
```

```
[1] 0 0
```

```
origin(b)
```

```
[1] 0.1382979 0.0000000
```

```
origin(w)
```

```
[1] 0.1382979 0.0000000
```

```
res(a)
```

```
[1] 0.5 0.5
```

```
res(b)
```

```
[1] 0.3297872 0.1612903
```

```
res(w)
```

```
[1] 0.3297872 0.1612903
```

# Revisiting Projections

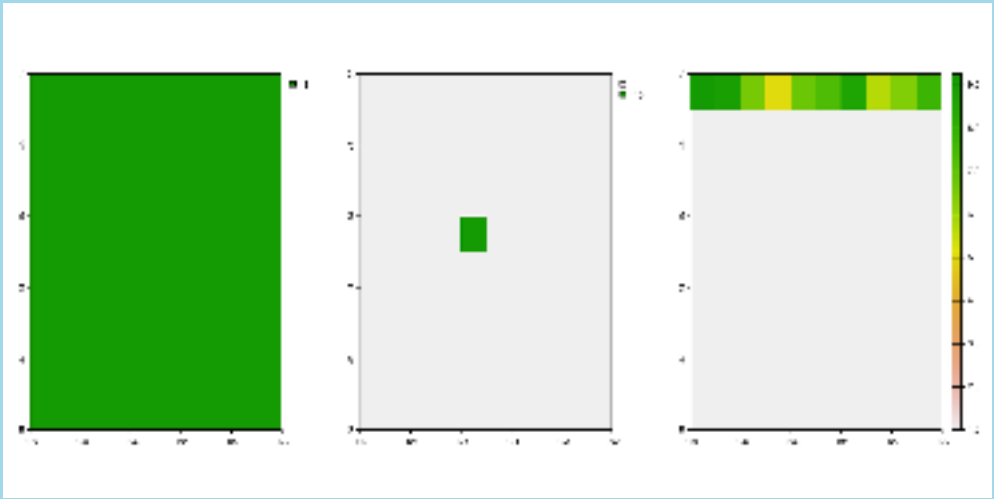
- if origin and extent are the same consider using `aggregate`, `disagg`, `extend` or `crop`

# Cell-wise operations

# Accessing Cell Values

- We can extract or change cell values using [ ]

```
a <- rast(ncols=10, nrows=10, xmin=-110, xmax=-100, ymin=40, ymax=50,
          crs="+proj=longlat +datum=WGS84")
values(a) <- 1
b1 <- a
b2 <- a
b1[5,5] <- 10
b2[1, 1:10 ] <- runif(10,4,10)
```

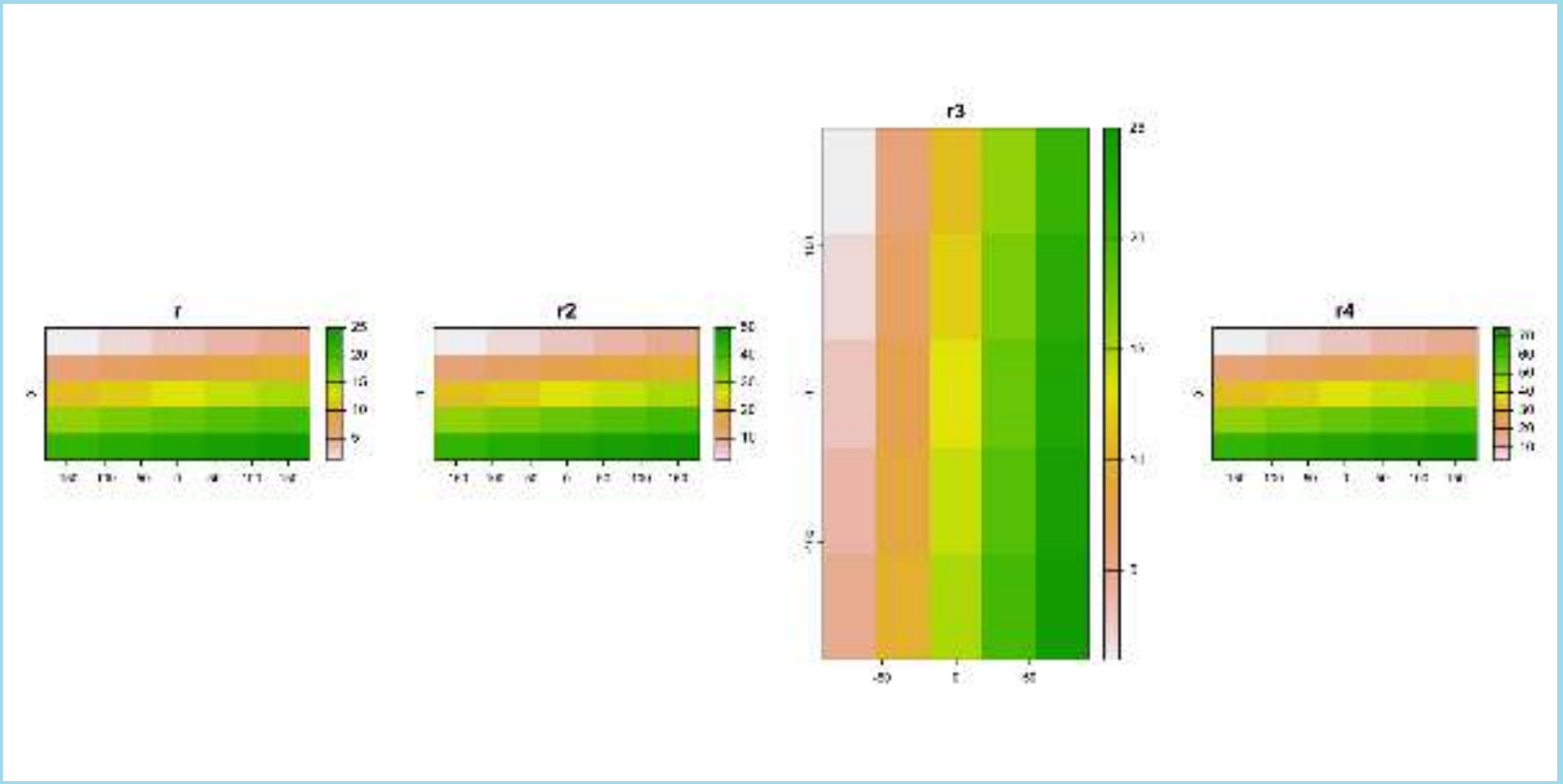


# Raster Math

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

# Raster Math

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

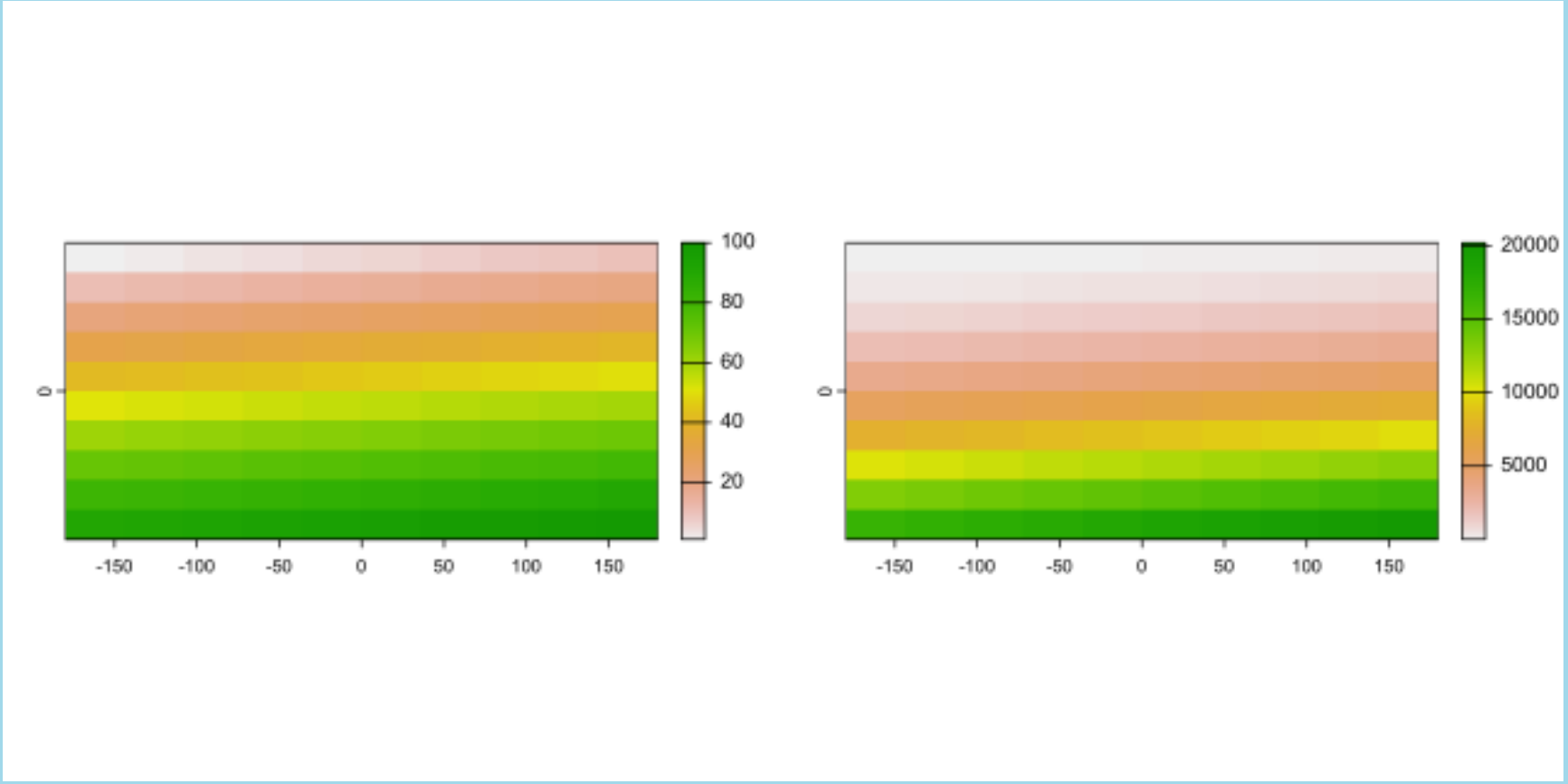


# Cell-wise operations

- `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

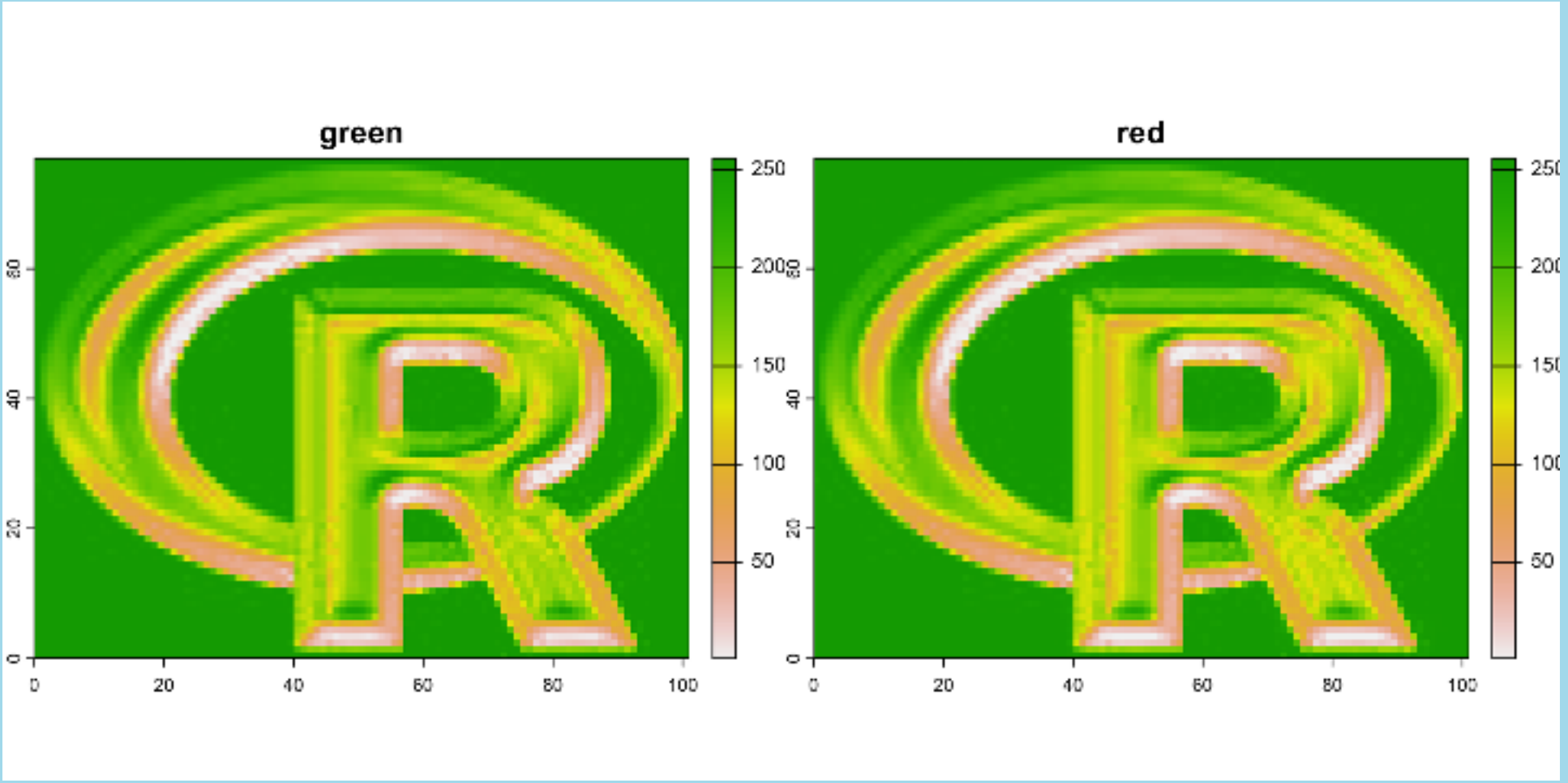
# Cell-wise operations

```
r <- rast(ncols=10, nrows=10)
values(r) <- 1:ncell(r)
f <- function(i) (i+1) * 2 * i + sqrt(i)
s <- app(r, f)
```



# Cell-wise Operations

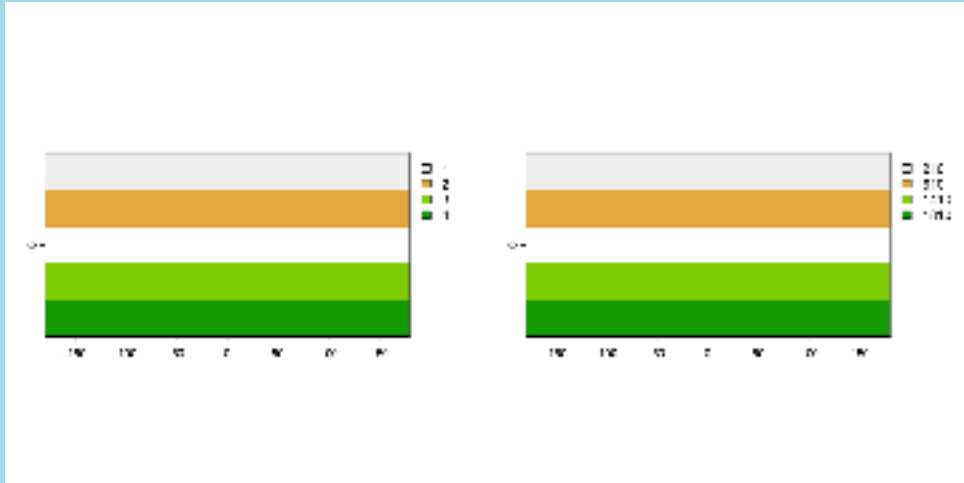
```
s <- rast(system.file("ex/logo.tif", package="terra")) + 1
ss <- s[[2:1]]

fvi <- function(x, y){ (x - y ) / (x + y) }
x <- lapp(ss, fun=fvi)
```

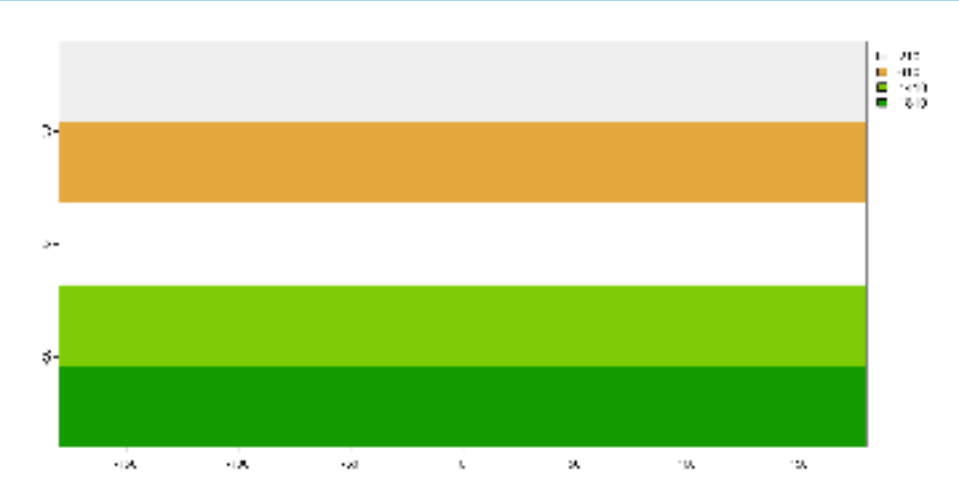

green
red
0
20
40
60
80
100
50
150
200
250

# Global Methods

- Provide summaries of 1 or more layers
- Use `zonal` to extract values from one layer based on `categorical` layer

```
1  r <- rast(ncols=10, nrows=10)
2  values(r) <- 1:ncell(r)
3  z <- rast(r)
4  values(z) <- rep(c(1:2, NA, 3:4),
5  names(z) <- "zone"
6  a <-  zonal(r, z, "sum", na.rm=TRU
7  b <- zonal(r, z, "sum", na.rm=TRUE
8  plot(b)
```





# Context-specific Functions

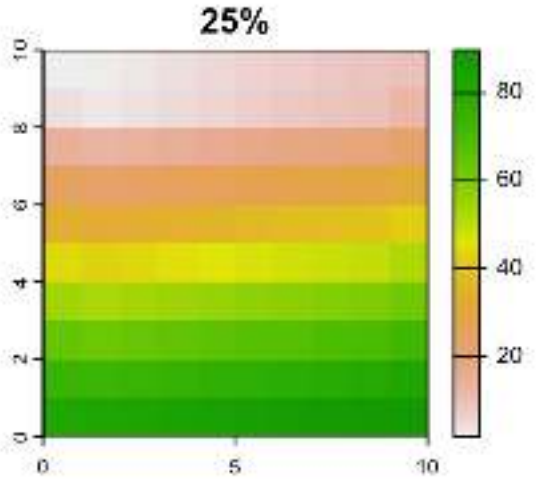
- `distance` and relatives are based on relationships between cells
- `focal` provides moving windows for smoothing data
- `terrain` allows calculation of slope, ruggedness, aspect using elevation rasters
- `shade` calculates hillshade based on terrain

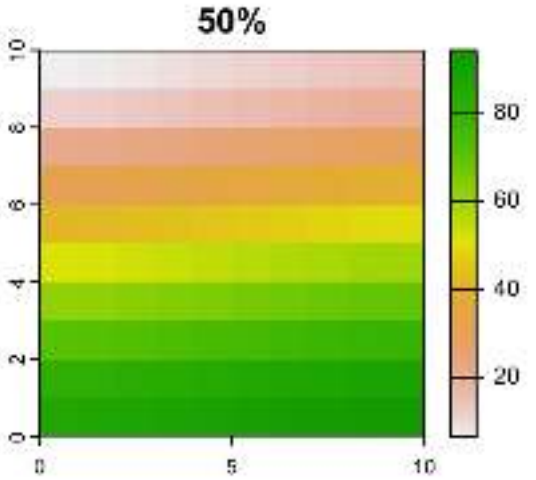
# Using `focal`

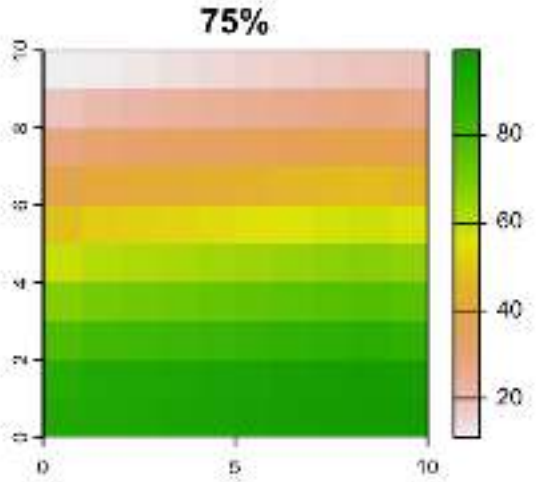
- `focal` requires a window (`w`) or weights matrix
- `na.policy` determines how to deal with `NA`s in the smoother
- `fillvalue` and `expand` tell `terra` what to do at the edges

```
r <- rast(ncols=10, nrows=10, ext(0, 10, 0, 10))
values(r) <- 1:ncell(r)

f <- focal(r, w=3, fun=function(x, ...) quantile(x, c(.25, .5, .75), ...),
plot(f)
```

# Using focal


25%
0
2
4
6
8
10
0
5
10
20
40
60
80


50%
0
2
4
6
8
10
0
5
10
20
40
60
80


75%
0
2
4
6
8
10
0
5
10
20
40
60
80