Spatial Data as Matrices and Rasters

HES 505 Fall 2022: Session 10

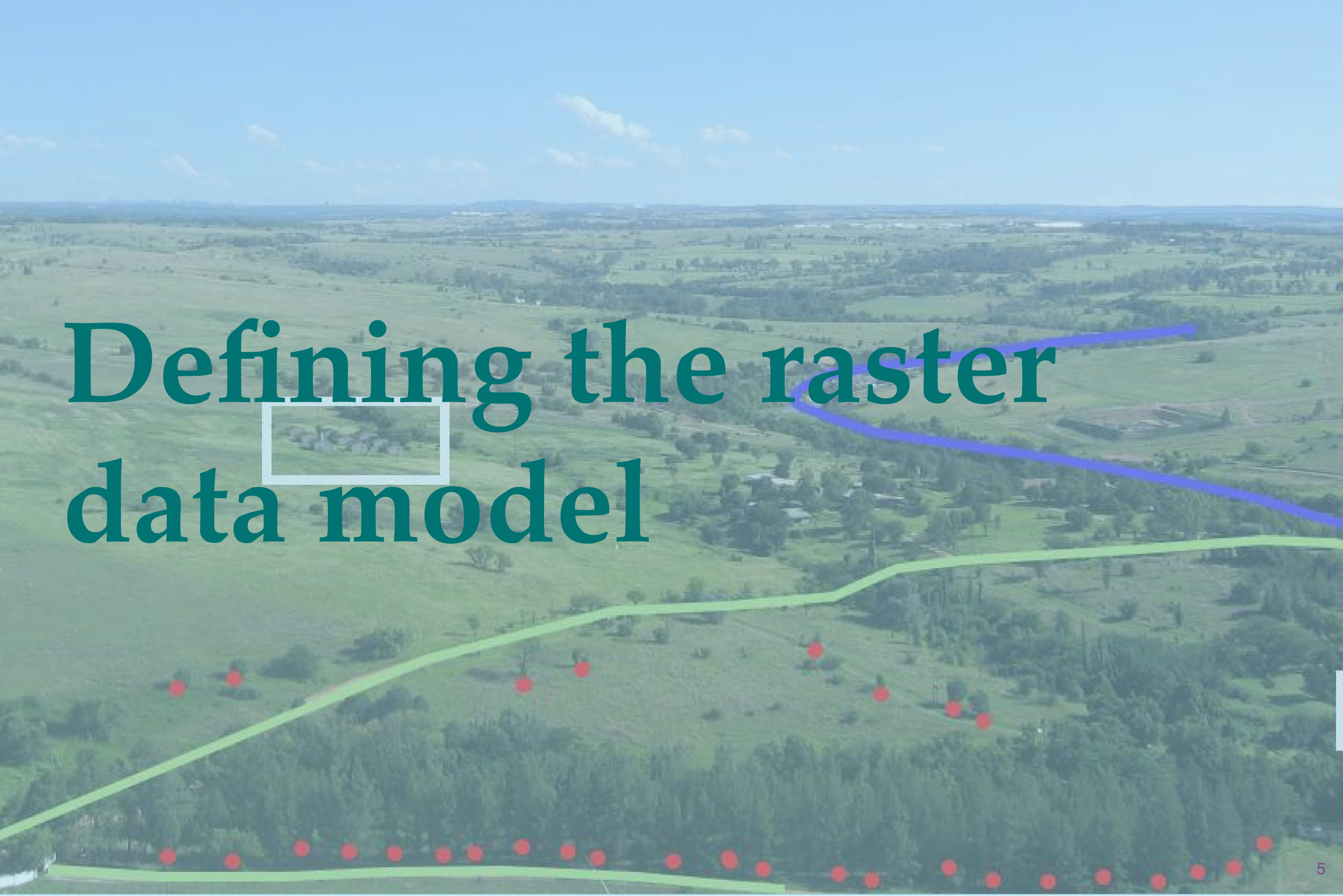
Matt Williamson

Today's Plan

Objectives

- By the end of today, you should be able to:
  - Describe the raster data model and its representation in R
  - Access the elements that define a raster
  - Build rasters from scratch using matrix operations and `terra`

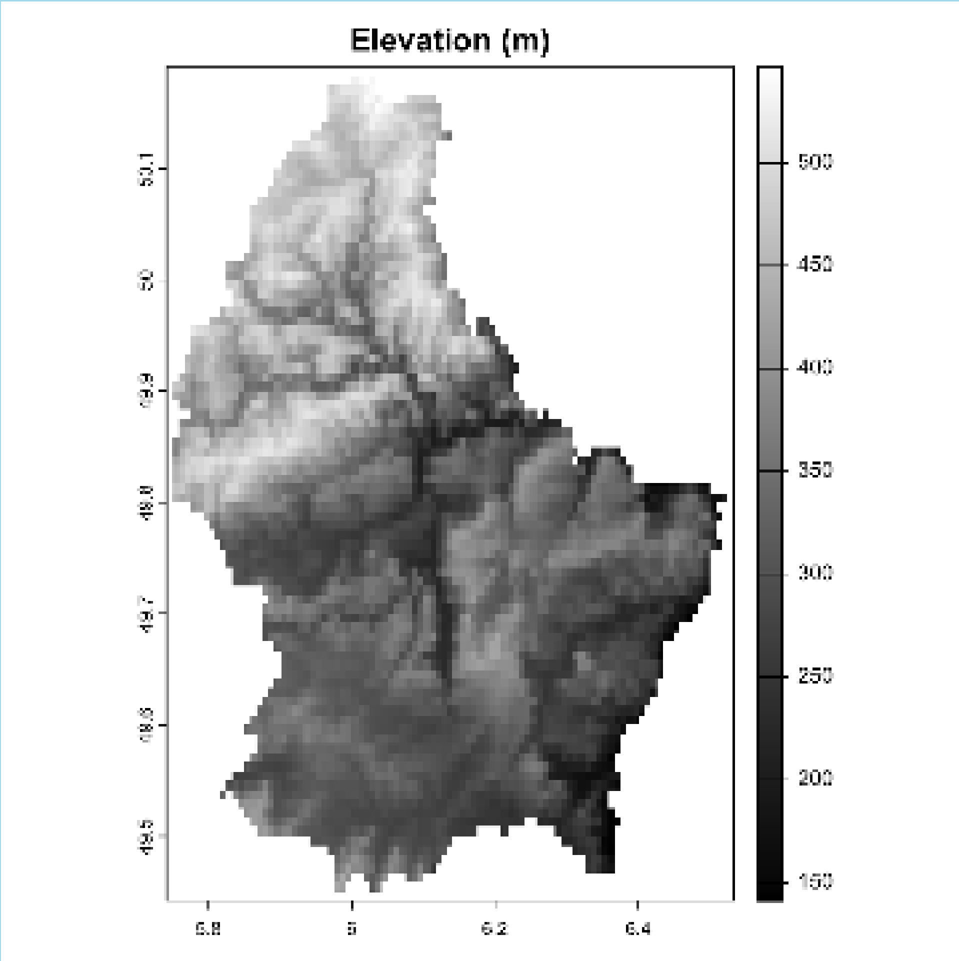
Defining the raster data model



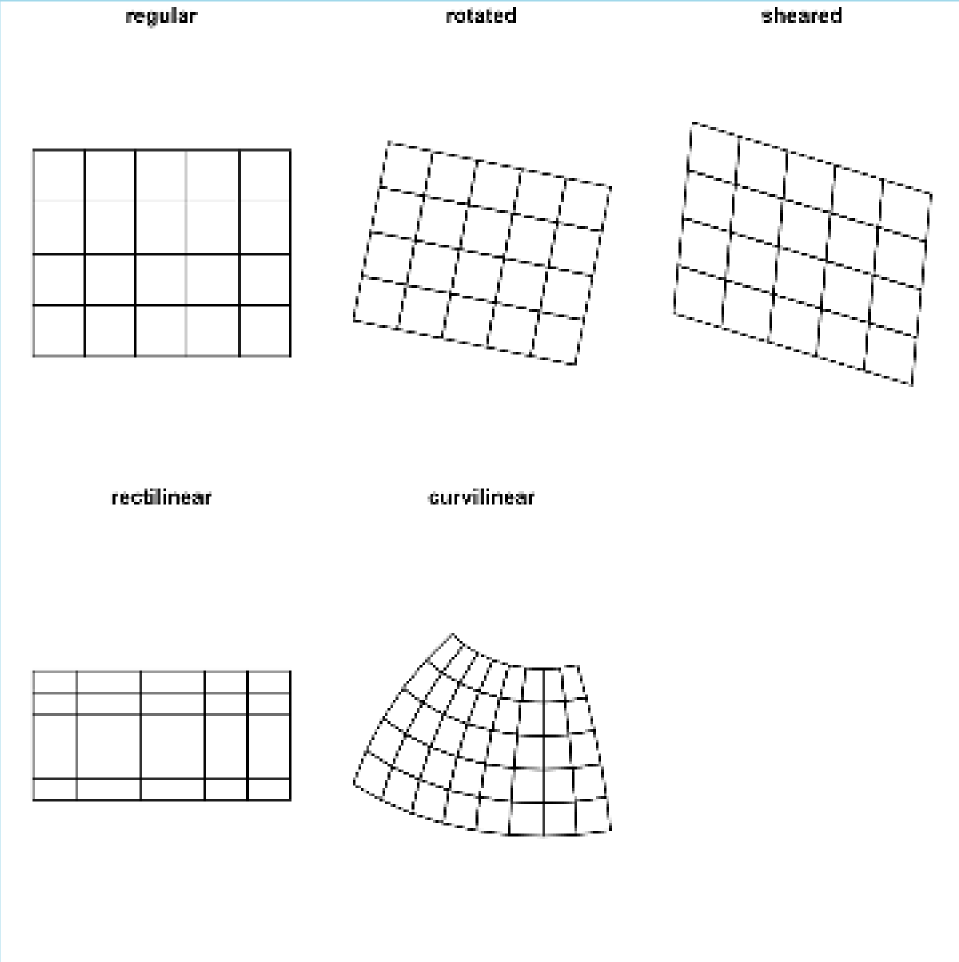
The Raster Data Model

Defining the Raster Data Model

- **Vector data** describe the "exact" locations of features on a landscape (including a Cartesian landscape)
- **Raster data** represent spatially continuous phenomena (`NA` is possible)
- Depict the alignment of data on a regular lattice (often a square)
  - Operations mimic those for `matrix` objects in `R`
- Geometry is implicit; the spatial extent and number of rows and columns define the cell size



Types of Raster Data



- **Regular**: constant cell size; axes aligned with Easting and Northing
- **Rotated**: constant cell size; axes not aligned with Easting and Northing
- **Sheared**: constant cell size; axes not parallel
- **Rectilinear**: cell size varies along a dimension
- **Curvilinear**: cell size and orientation dependent on the other dimension

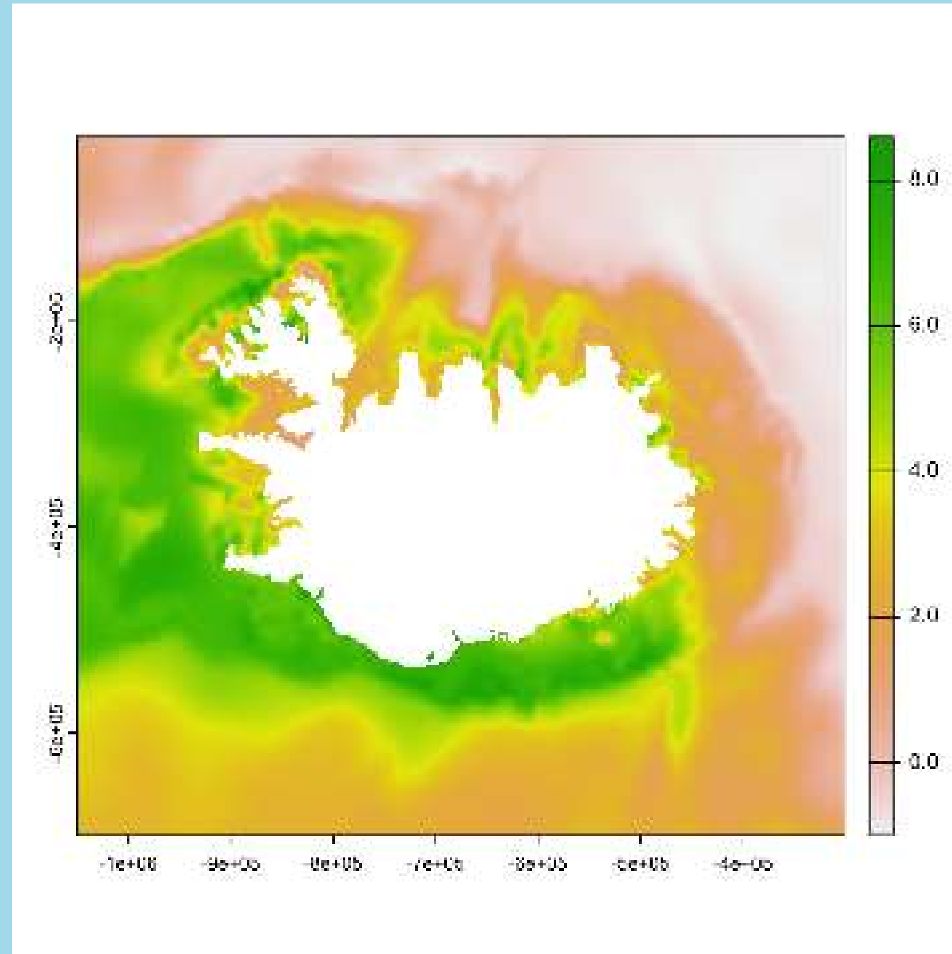
Types of Raster Data

- **Continuous**: numeric data representing a measurement (e.g., elevation, precipitation)
- **Categorical**: integer data representing factors (e.g., land use, land cover)

Continuous Rasters

```r
mintemp <- rast("ftp://ftp.hafro.i
mintemp
```

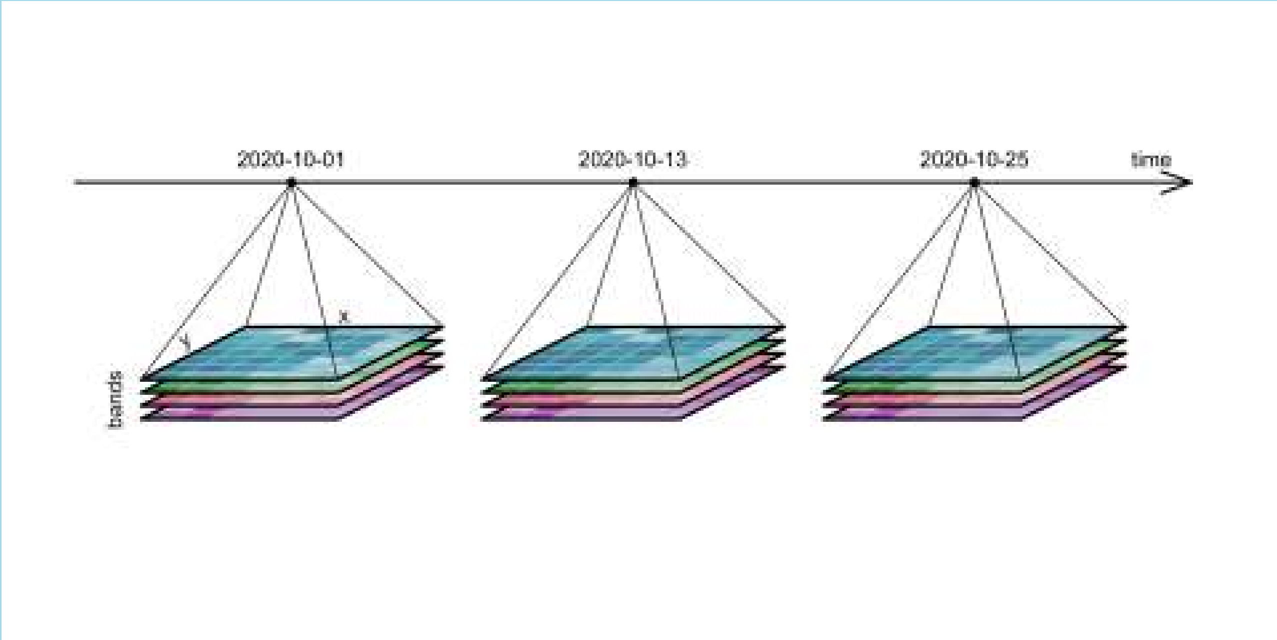
```
class       : SpatRaster
dimensions  : 340, 375, 1  (nrow,
ncol, nlyr)
resolution  : 2000, 2000  (x, y)
extent      : -1050226, -300225.9,
-699984.3, -19984.32  (xmin, xmax,
ymin, ymax)
coord. ref. : +proj=laea +lat_0=69
+lon_0=-4 +x_0=0 +y_0=0 +datum=WGS84
+units=m +no_defs
source      : Iceland_minbtemp.tif
name        : Iceland_minbtemp
min value   :       -0.9982879
max value   :        8.6031137
```



Categorical Rasters

Adding Dimensions

- When data are aligned in space and/or time, more efficient to process as ‘cubes’ or ‘stacks’
- Bands of satellite imagery, multiple predictors, spatio-temporal data



A note about support

- We talked briefly about the `agr` option in the `st_sf()` function
- `agr` refers to the attribute-geometry-relationship which can be:
    - **constant** = applies to every point in the geometry (lines and polygons are just lots of points)
    - **identity** = a value unique to a geometry
    - **aggregate** = a single value that integrates data across the geometry
- **Support** is the area to which an attribute applies.
- Rasters can have **point** (attribute refers to the cell center) or **cell** (attribute refers to an area similar to the pixel) support

Rasters in R

Rasters in R

- `raster` - the original workhorse package; built on `sp`, `rgdal`, and `rgeos`
  - `RasterLayer`, `RasterStack`, and `RasterBrick` classes
- `terra` - relatively new; developed by the `raster` folks, but designed to be much faster
  - `SpatRaster` and `SpatVector` classes
- `stars` - developed by `sf` package developers; `tidyverse` compatible; designed for spatio-temporal data
  - `stars` class
  - Crosswalk between `raster` and `stars` is available here
  - Only way to deal with *rectilinear* and *curvilinear* data

Rasters with `terra`

- syntax is different for `terra` compared to `sf`
- Representation in `Environment` is also different
- Can break pipes, **Be Explicit**

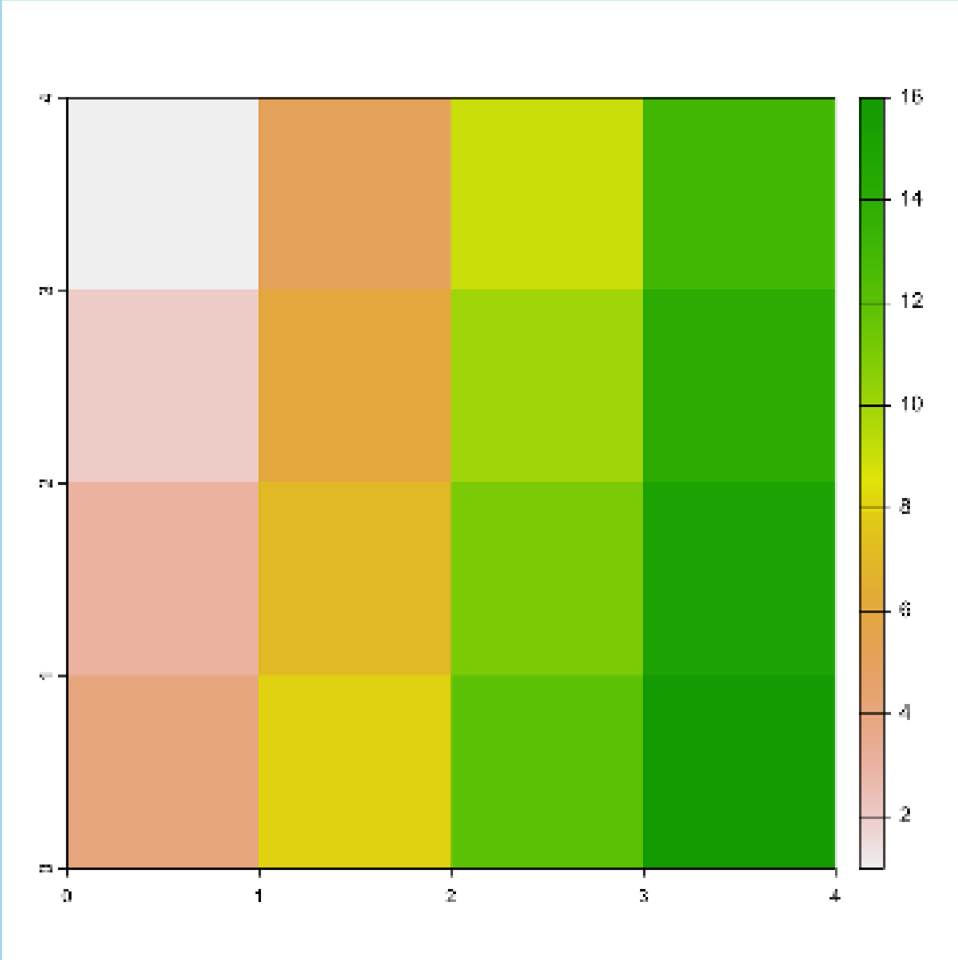
Rasters by Construction

```r
mtx <- matrix(1:16, nrow=4)
mtx
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

```r
rstr <- terra::rast(mtx)
rstr
```

```
class       : SpatRaster
dimensions  : 4, 4, 1  (nrow, ncol,
nlyr)
resolution  : 1, 1  (x, y)
extent      : 0, 4, 0, 4  (xmin,
xmax, ymin, ymax)
coord. ref. :
source(s)   : memory
name        : lyr.1
min value   :     1
max value   :    16
```



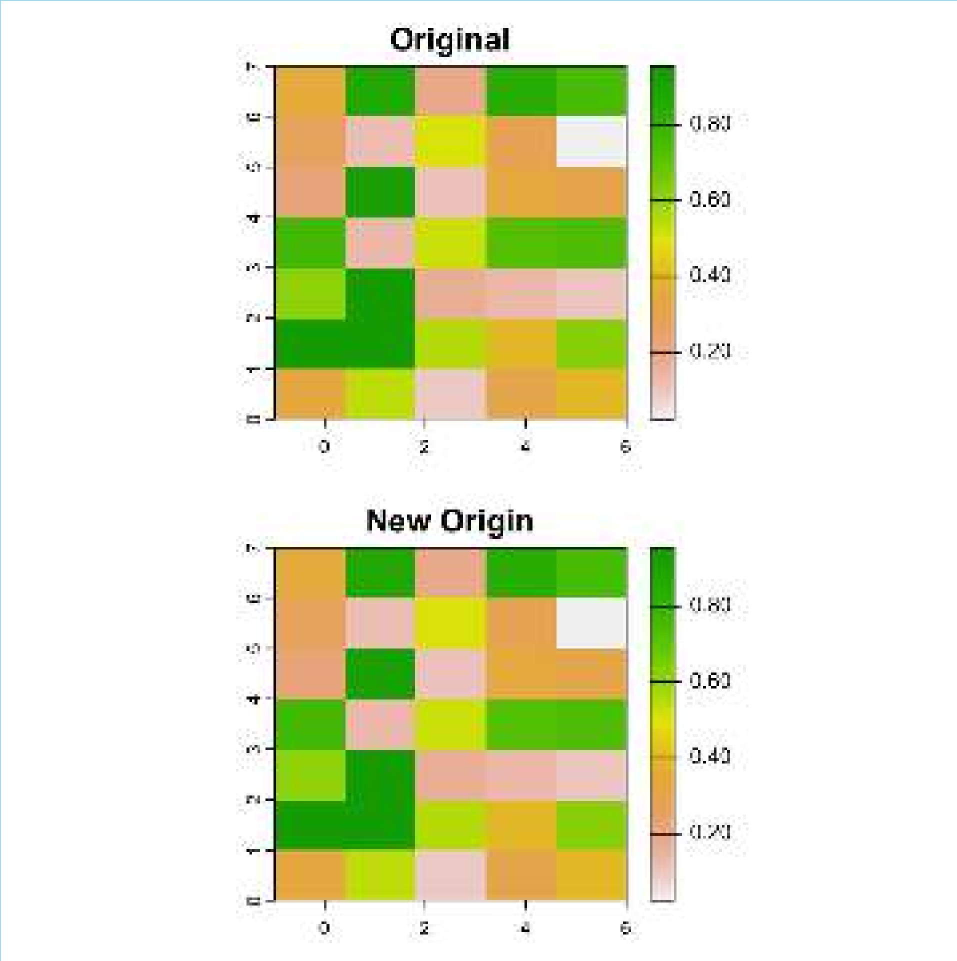
Rasters by Construction: Origin

- Origin defines the location of the intersection of the x and y axes

```r
r <- rast(xmin=-4, xmax = 9.5, ncc
r[] <- runif(ncell(r))
origin(r)
```

```
[1] 0.05 0.00
```

```r
r2 <- r
origin(r2) <- c(2,2)
```



Rasters by Construction: Resolution

- Geometry is implicit; the spatial extent and number of rows and columns define the cell size
- **Resolution** (`res`) defines the length and width of an individual pixel

```
r <- rast(xmin=-4, xmax = 9.5,
          ncols=10)
res(r)
```

```
[1] 1.35 1.00
```

```
r2 <- rast(xmin=-4, xmax = 5,
           ncols=10)
res(r2)
```

```
[1] 0.9 1.0
```

```
r <- rast(xmin=-4, xmax = 9.5,
          res=c(0.5,0.5))
ncol(r)
```

```
[1] 27
```

```
r2 <- rast(xmin=-4, xmax = 9.5,
           res=c(5,5))
ncol(r2)
```

```
[1] 3
```

Rasters from Files

- Building rasters useful for templates
- More common to read from files

```
r <- rast(system.file("ex/elev.tif", package="terra"))
r
```

```
class       : SpatRaster
dimensions  : 90, 95, 1  (nrow, ncol, nlyr)
resolution  : 0.008333333, 0.008333333  (x, y)
extent      : 5.741667, 6.533333, 49.44167, 50.19167  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : elev.tif
name        : elevation
min value   :       141
max value   :       547
```

Accessing Raster Attributes: Coordinate Reference System

- `terra` stores CRS in WKT format
- Can set and access using `EPSG` and `proj` (deprecated)
- Pay attention to case

```r
r <- rast(system.file("ex/elev.tif", package="terra"))
crs(r)
```

```
[1] "GEOGCRS[\"WGS 84\",\n    DATUM[\"World Geodetic System 1984\",\n
ELLIPSOID[\"WGS 84\",6378137,298.257223563,\n
LENGTHUNIT[\"metre\",1]]],\n    PRIMEM[\"Greenwich\",0,\n
ANGLEUNIT[\"degree\",0.0174532925199433]],\n    CS[ellipsoidal,2],\n
AXIS[\"geodetic latitude (Lat)\",north,\n            ORDER[1],\n
ANGLEUNIT[\"degree\",0.0174532925199433]],\n        AXIS[\"geodetic longitude
(Lon)\",east,\n            ORDER[2],\n
ANGLEUNIT[\"degree\",0.0174532925199433]],\n    ID[\"EPSG\",4326]]"
```

Accessing Raster Attributes: Coordinate Reference System

```r
r <- rast(system.file("ex/elev.tif", package="terra"))
crs(r, describe=TRUE)
```

```
    name authority code area          extent
1 WGS 84      EPSG 4326 <NA> NA, NA, NA, NA
```

```r
crs(r, proj=TRUE)
```

```
[1] "+proj=longlat +datum=WGS84 +no_defs"
```

```r
crs(r, parse=TRUE)
```

```
 [1] "GEOGCRS[\"WGS 84\","
 [2] "    DATUM[\"World Geodetic System 1984\","
 [3] "        ELLIPSOID[\"WGS 84\",6378137,298.257223563,"
 [4] "            LENGTHUNIT[\"metre\",1]]],"
 [5] "    PRIMEM[\"Greenwich\",0,"
 [6] "        ANGLEUNIT[\"degree\",0.0174532925199433]],"
 [7] "    CS[ellipsoidal,2],"
 [8] "        AXIS[\"geodetic latitude (Lat)\",north,"
 [9] "            ORDER[1],"
[10] "            ANGLEUNIT[\"degree\",0.0174532925199433]],"
[11] "        AXIS[\"geodetic longitude (Lon)\",east,"
```

```
[12] "                ORDER[2],"
[13] "                ANGLEUNIT[\"degree\",0.0174532925199433]],"
[14] "    ID[\"EPSG\",4326]]"
```

Accessing Raster Attributes: Bounding box

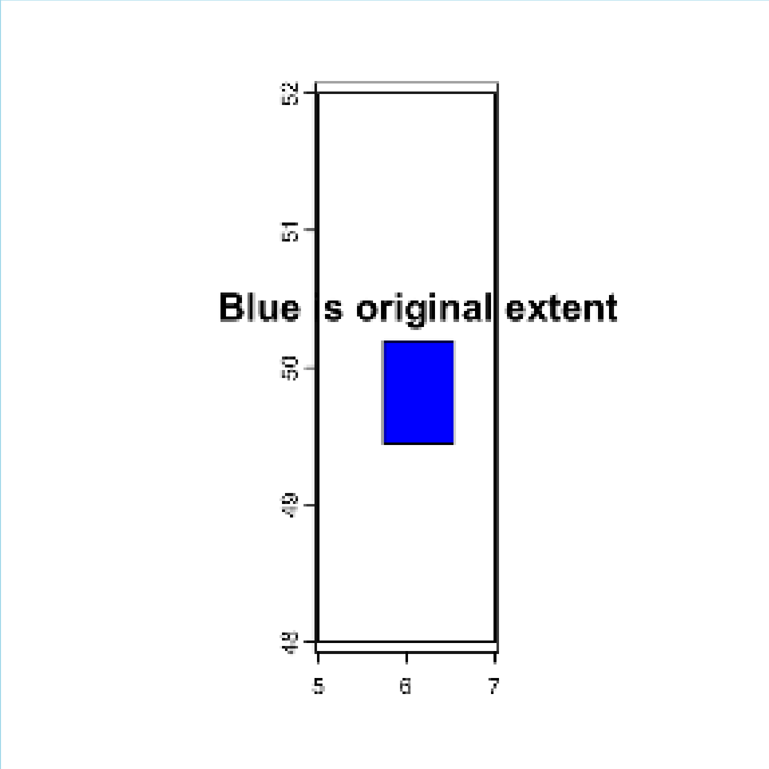
- `terra` uses `ext()` to get or set the extent/bounding box
- Fills cells with `NA`

```r
ext(r)
```

```
SpatExtent : 5.74166666666667, 6.53333333333333, 49.4416666666667, 50.1916666666667 (xmin, xmax, ymin, ymax)
```

```r
r2 <- r
ext(r2) <- c(5, 7, 48, 52)
ext(r2)
```

```
SpatExtent : 5, 7, 48, 52 (xmin, xmax, ymin, ymax)
```



Converting vectors to rasters

- Sometimes necessary to convert between data models
- `raster::rasterize`, `terra::rasterize`, `stars::st_rasterize`, and `fasterize::fasterize` all will convert polygons to raster data
- `stars::st_polygonize` will work in the opposite direction
- `terra::vect` will read in vectors as `SpatVectors` or coerce `sf` to `SpatVector`