

Spatial Data as Matrices and Rasters

HES 505 Fall 2022: Session 10

Matt Williamson



Today's Plan

Objectives

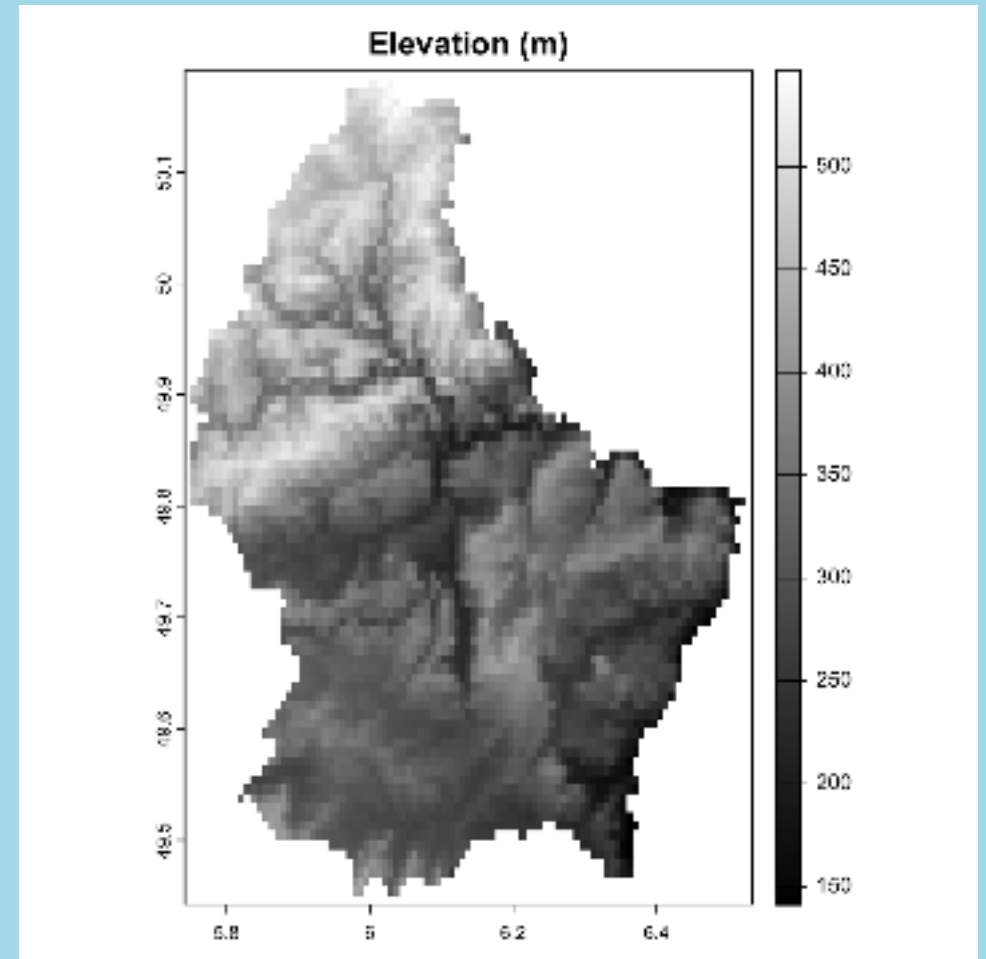
- By the end of today, you should be able to:
 - Describe the raster data model and its representation in R
 - Access the elements that define a raster
 - Build rasters from scratch using matrix operations and **terra**

Defining the raster data model

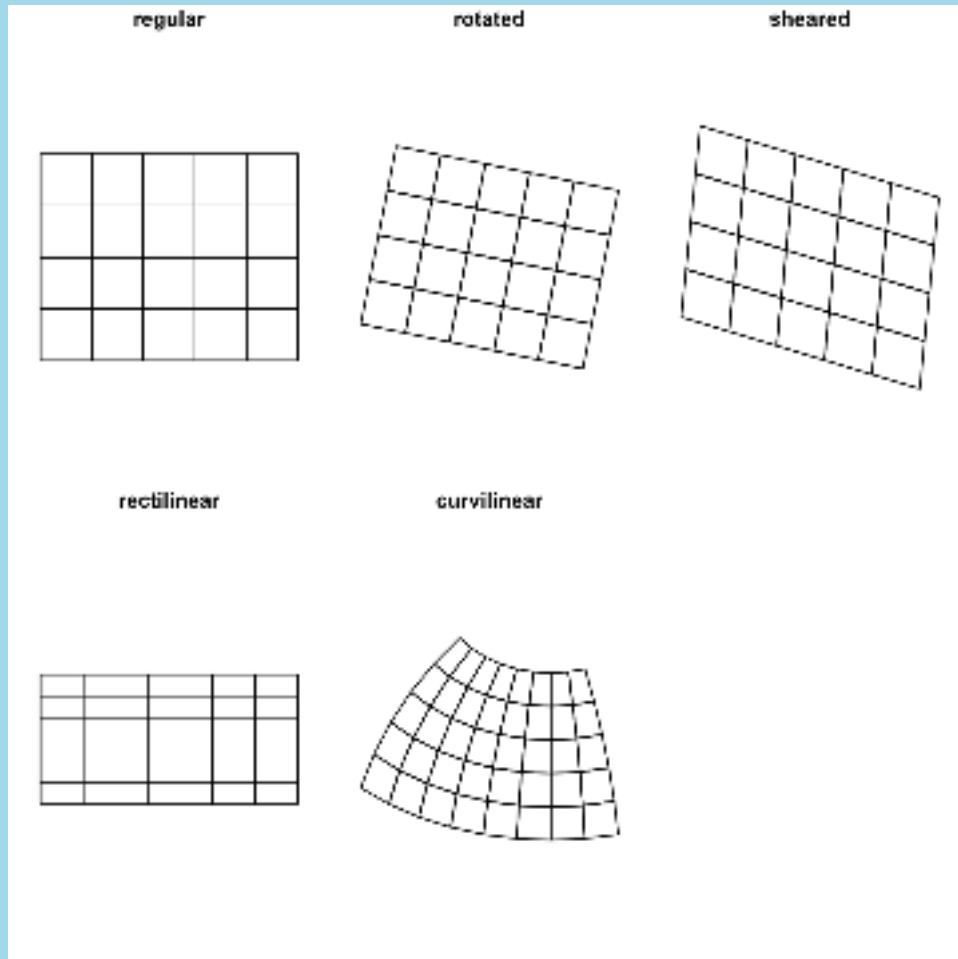
The Raster Data Model

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

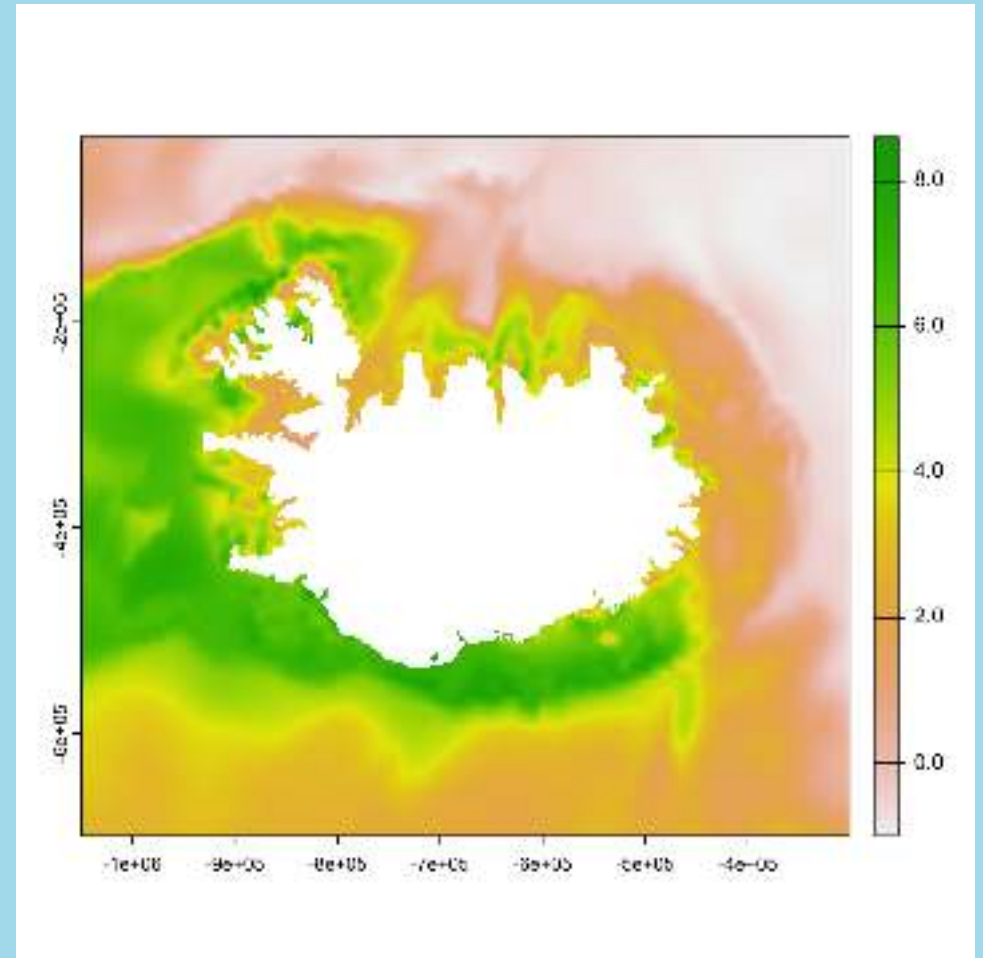
Types of Raster Data

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

Continuous Rasters

```
1 mintemp <- rast("ftp://ftp.hafro.is/ftp/pub/vegis/vegdata/1990-1999/1990-1999_mintemp.tif")
2 mintemp
```

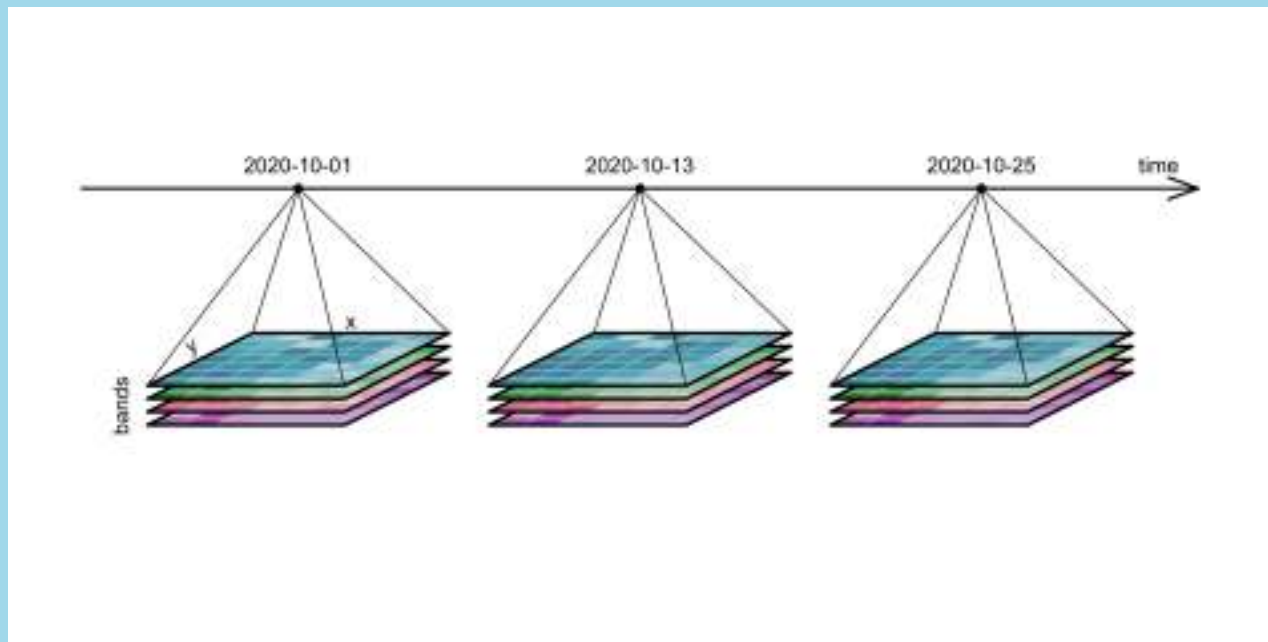
```
class           : SpatRaster
dimensions      : 340, 375, 1  (nrow,
ncol, nlyr)
resolution     : 2000, 2000  (x, y)
extent          : -1050226, -300225.9,
-699984.3, -19984.32  (xmin, xmax,
ymin, ymax)
coord. ref.    : +proj=laea +lat_0=69
+lon_0=-4 +x_0=0 +y_0=0 +datum=WGS84
+units=m +no_defs
source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

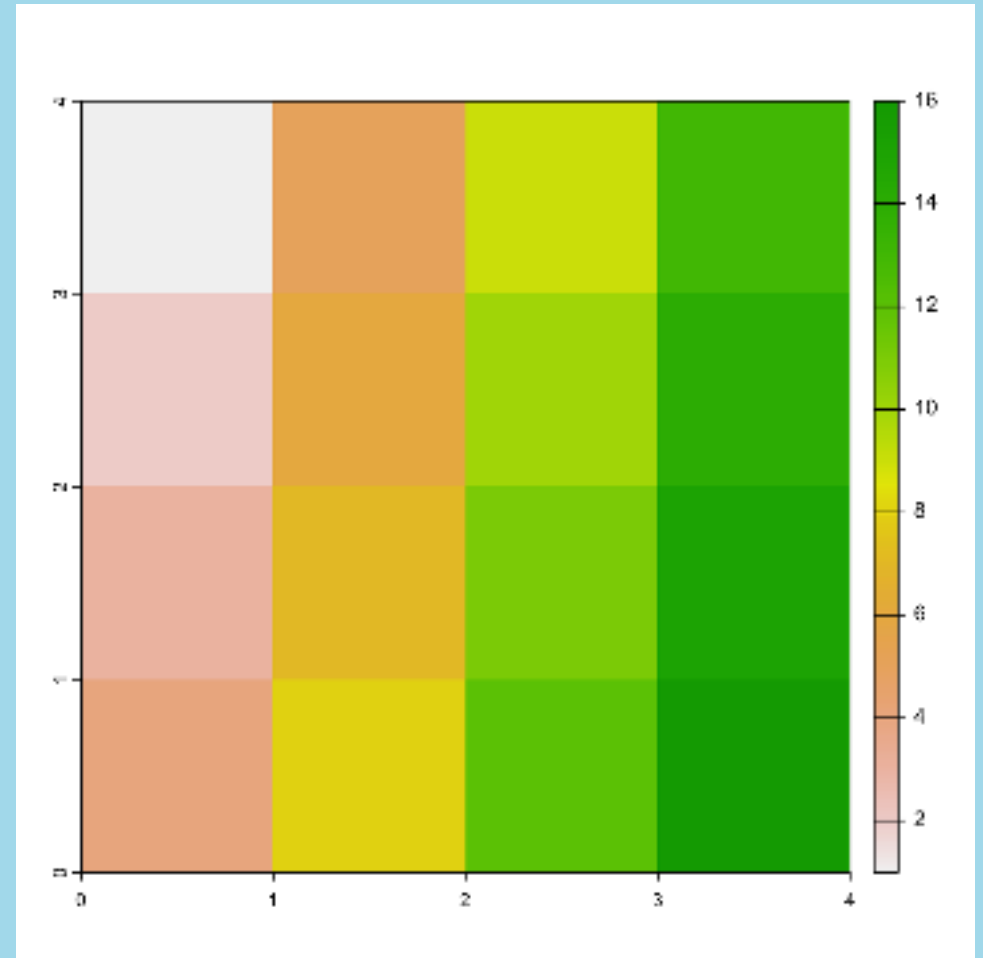
Rasters by Construction

```
1 mtx <- matrix(1:16, nrow=4)
2 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
```

```
1 rstr <- terra::rast(mtx)
2 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
```



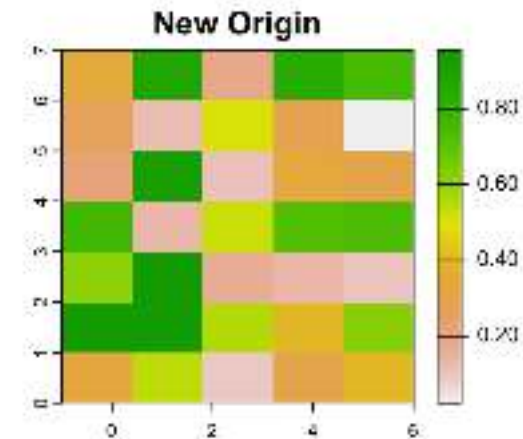
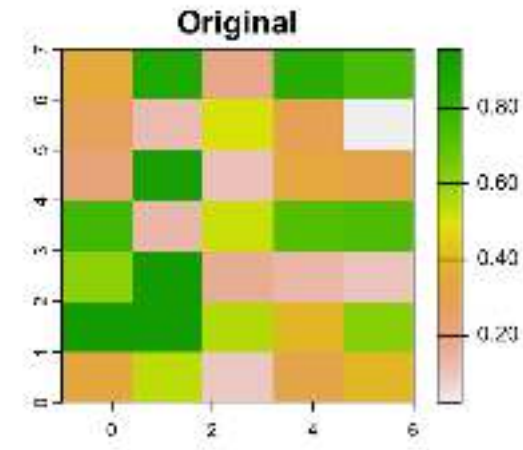
Rasters by Construction: Origin

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

```
1 r <- rast(xmin=-4, xmax = 9.5, ncol=10, nrow=10)
2 r[] <- runif(ncell(r))
3 origin(r)
```

```
[1] 0.05 0.00
```

```
1 r2 <- r
2 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

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

```
[1] 1.35 1.00
```

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

```
[1] 0.9 1.0
```

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

```
[1] 27
```

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

```
[1] 3
```

Rasters from Files

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

```
1 r <- rast(system.file("ex/elev.tif", package="terra"))  
2 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

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

```
[1] "GEOGCRS[\"WGS 84\", \n      ENSEMBLE[\"World Geodetic System 1984\nensemble\", \n      MEMBER[\"World Geodetic System 1984 (Transit)\", \nMEMBER[\"World Geodetic System 1984 (G730)\", \n      MEMBER[\"World\nGeodetic System 1984 (G873)\", \n      MEMBER[\"World Geodetic System 1984\n(G1150)\", \n      MEMBER[\"World Geodetic System 1984 (G1674)\", \nMEMBER[\"World Geodetic System 1984 (G1762)\", \n      MEMBER[\"World\nGeodetic System 1984 (G2139)\", \n      ELLIPSOID[\"WGS\n84\", 6378137, 298.257223563, \n      LENGTHUNIT[\"metre\", 1]], \nENSEMBLEACCURACY[2.0]], \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\n      (Lon)",east,\n        ORDER[2],\n      ANGLEUNIT["degree",0.0174532925199433]],\n      USAGE[\n      SCOPE["Horizontal component of 3D system."],\n      AREA["World."],\n
```

Accessing Raster Attributes: Coordinate Reference System

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

```
      name authority code  area      extent
1 WGS 84      EPSG 4326 World -180, 180, 90, -90
```

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

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

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

```
[1] "GEOGCRS[\"WGS 84\", \"
[2] \"      ENSEMBLE[\"World Geodetic System 1984 ensemble\", \"
[3] \"      MEMBER[\"World Geodetic System 1984 (Transit)\", \"
[4] \"      MEMBER[\"World Geodetic System 1984 (G730)\", \"
[5] \"      MEMBER[\"World Geodetic System 1984 (G873)\", \"
[6] \"      MEMBER[\"World Geodetic System 1984 (G1150)\", \"
[7] \"      MEMBER[\"World Geodetic System 1984 (G1674)\", \"
[8] \"      MEMBER[\"World Geodetic System 1984 (G1762)\", \"
[9] \"      MEMBER[\"World Geodetic System 1984 (G2139)\", \"
[10] \"      ELLIPSOID[\"WGS 84\", 6378137, 298.257223563, \"
[11] \"      LENGTHUNIT[\"metre\", 1]], \"
```

```
[12] "          ENSEMBLEACCURACY[2.0]],"  
[13] "    PRIMEM[\"Greenwich\",0,\"  
[14] "          ANGLEUNIT[\"degree\",0.0174532925199433]],"  
[15] "    CS[ellipsoidal,21,"
```


Accessing Raster Attributes: Bounding box

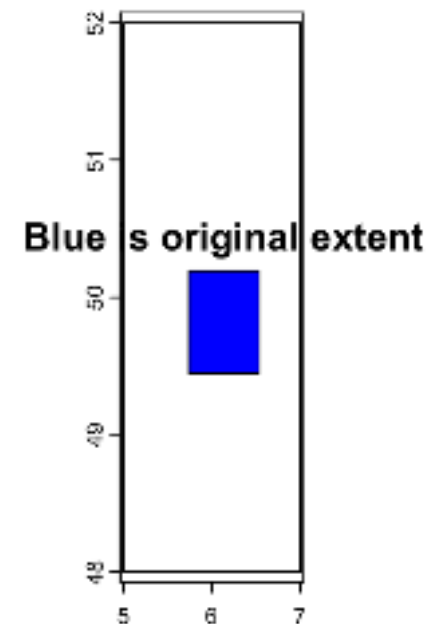
- **terra** uses **ext()** to get or set the extent/bounding box
- Fills cells with **NA**

```
1 ext(r)
```

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

```
1 r2 <- r  
2 ext(r2) <- c(5, 7, 48, 52)  
3 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`

