Operations on Raster Data I

HES 505 Fall 2024: Session 12

Carolyn Koehn

Today’s Plan

Objectives

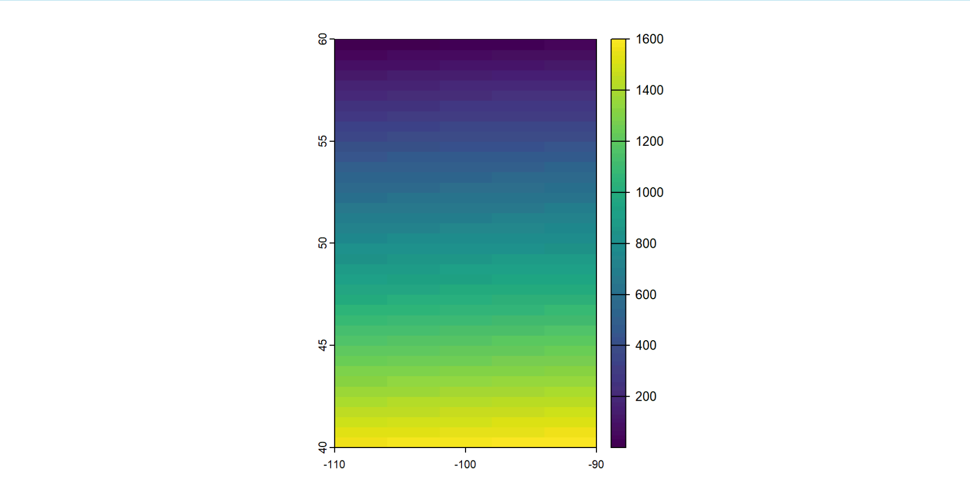
- By the end of today, you should be able to:
  - Align rasters for spatial processing
  - Adjust the resolution of raster data
  - Combine (or reduce) rasters to match the extent of your analysis

Aligning rasters for spatial processing

Projecting raster data

- Transformation from lat/long to planar CRS involves some loss of precision
- New cell values estimated using overlap with original cells
- Interpolation for continuous data, nearest neighbor for categorical data
- Equal-area projections are preferred; especially for large areas

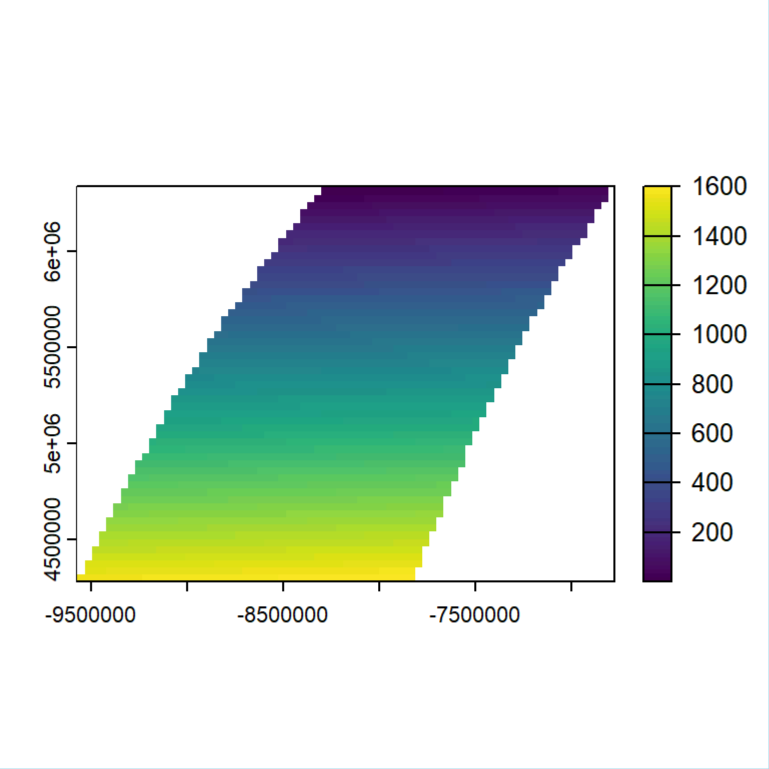
```r
library(sf)
library(terra)
library(spDataLarge)
r <- rast(xmin=-110, xmax=-90, ymin=40, ym
values(r) <- 1:ncell(r)
plot(r)
```



Projecting raster data

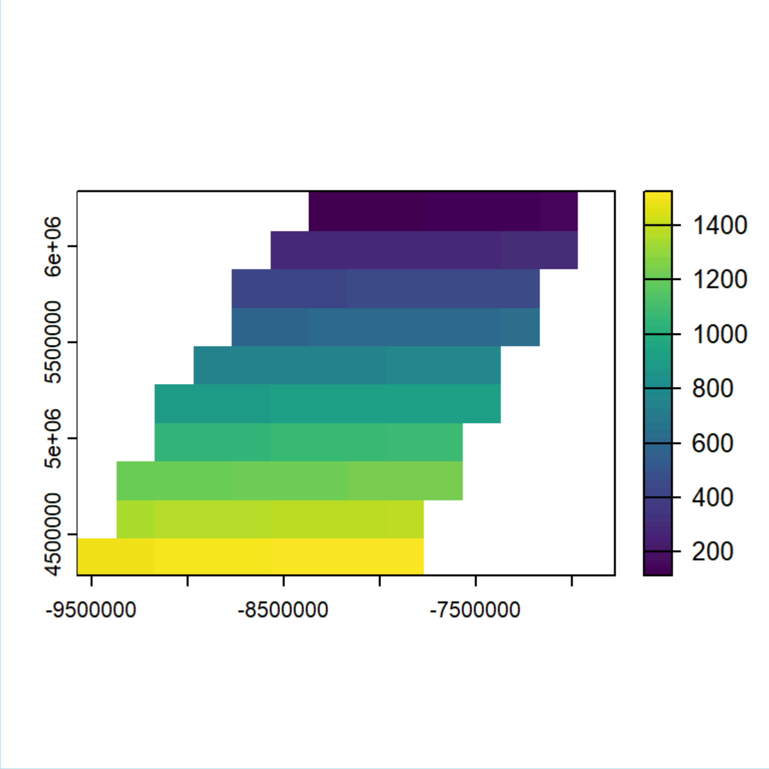
- simple method; alignment not guaranteed

```
newcrs <- "+proj=robin +da
pr1 <- terra::project(r,
plot(pr1)
```



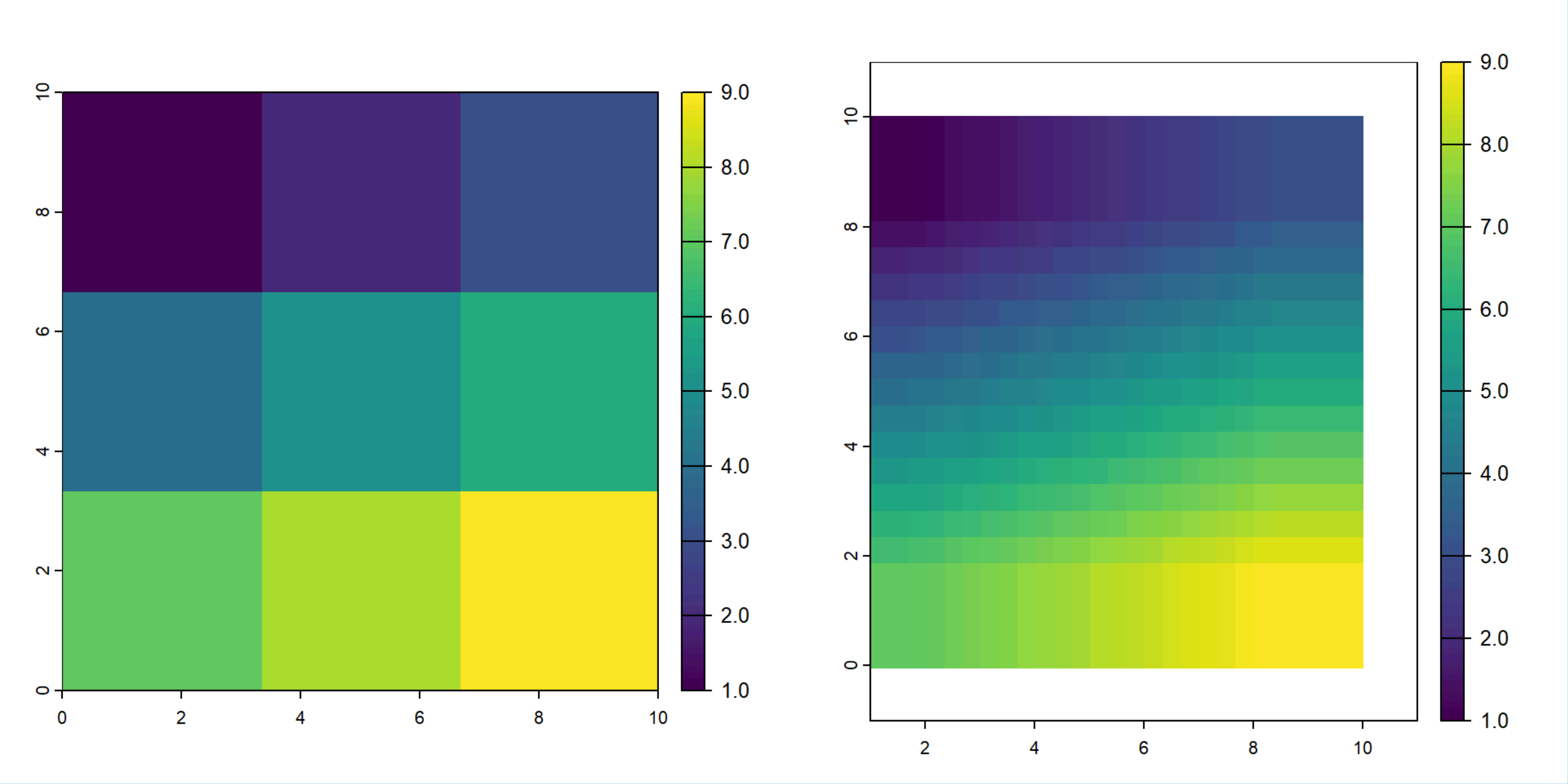
- providing a template to ensure alignment

```
x <- rast(pr1)
# Set the cell size
res(x) <- 200000
pr3 <- terra::project(r,
plot(pr3)
```



Aligning Data: `resample`

```r
r <- rast(nrow=3, ncol=3, xmin=0, xmax=10, ymin=0, ymax=10)
values(r) <- 1:ncell(r)
s <- rast(nrow=25, ncol=30, xmin=1, xmax=11, ymin=-1, ymax=11)
x <- resample(r, s, method="bilinear")
```



Adjusting resolution

Downscaling and Upscaling

- Aligning data for later analysis
- Remembering *scale*
- Thinking about support

Changing resolutions

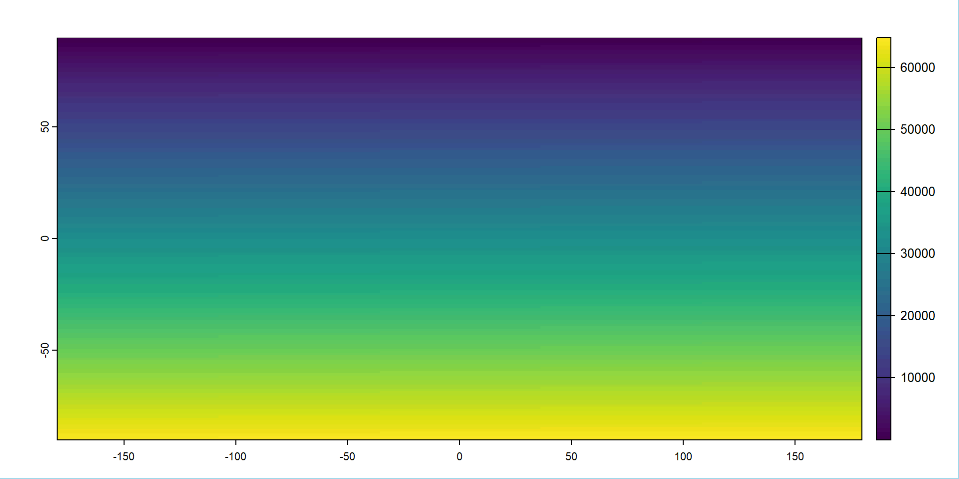
- `aggregate`, `disaggregate`, `resample` allow changes in cell size
- `aggregate` requires a function (e.g., `mean()` or `min()`) to determine what to do with the grouped values
- `resample` allows changes in cell size **and** shifting of cell centers (slower)

Changing resolutions: aggregate

```r
r <- rast()
r
```

```
class       : SpatRaster
dimensions  : 180, 360, 1  (nrow,
ncol, nlyr)
resolution  : 1, 1  (x, y)
extent      : -180, 180, -90, 90
(xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (CRS84)
(OGC:CRS84)
```

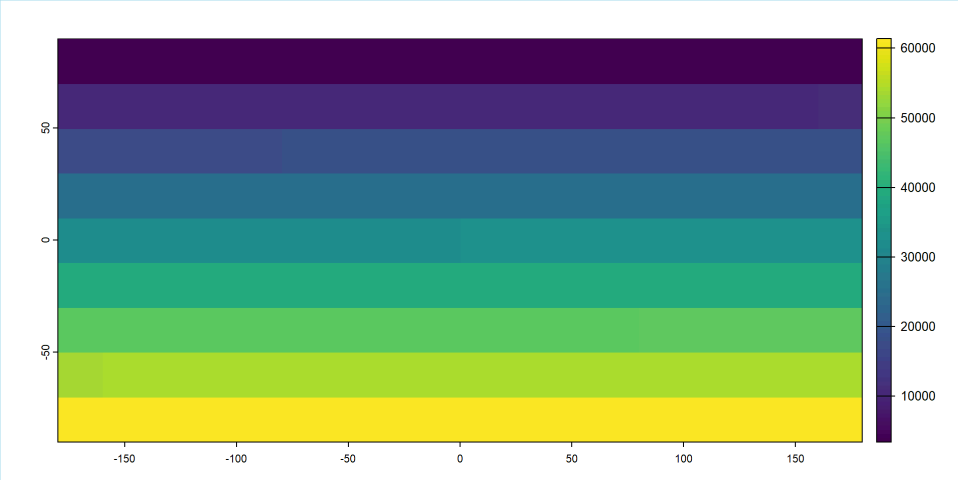
```r
values(r) <- 1:ncell(r)
plot(r)
```



```r
ra <- aggregate(r, 20)
ra
```

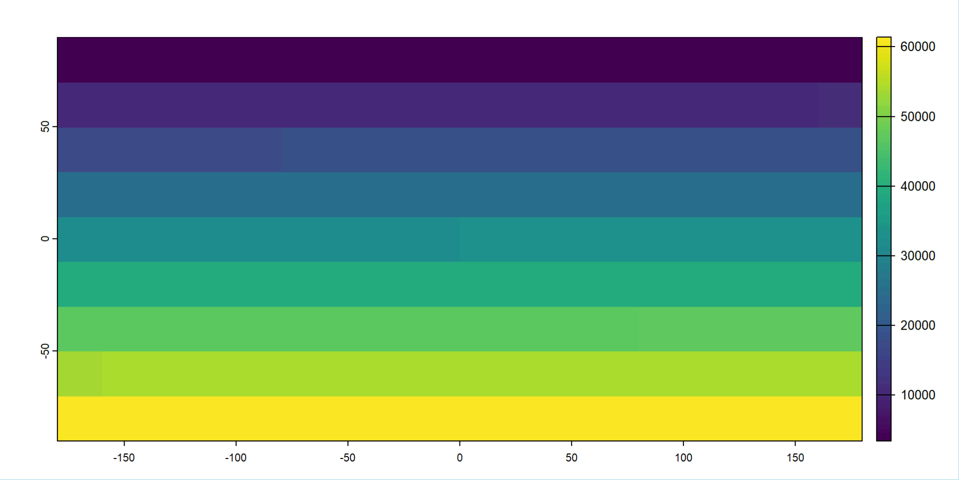
```
class       : SpatRaster
dimensions  : 9, 18, 1  (nrow, ncol,
nlyr)
resolution  : 20, 20  (x, y)
extent      : -180, 180, -90, 90
(xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (CRS84)
(OGC:CRS84)
source(s)   : memory
name        :   lyr.1
min value   :  3430.5
max value   : 61370.5
```

```r
plot(ra)
```



Changing resolutions: disagg

```r
ra <- aggregate(r, 20)
plot(ra)
```

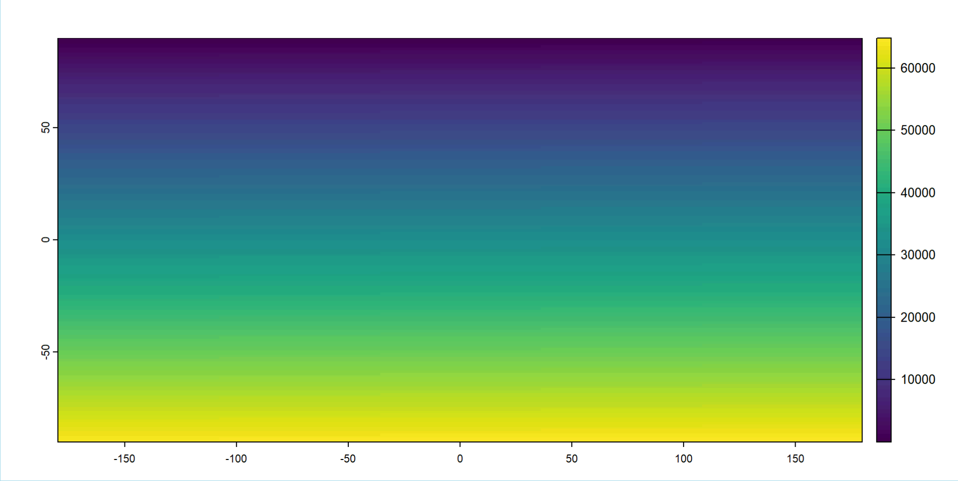


```r
rd <- disagg(r, 20)
```

```r
rd
```

```
class          : SpatRaster
dimensions     : 3600, 7200, 1  (nrow,
ncol, nlyr)
resolution     : 0.05, 0.05  (x, y)
extent         : -180, 180, -90, 90
(xmin, xmax, ymin, ymax)
coord. ref.    : lon/lat WGS 84 (CRS84)
(OGC:CRS84)
source         :
spat_27a03b4b79c4_10144.tif
name           : lyr.1
min value      :     1
max value      : 64800
```

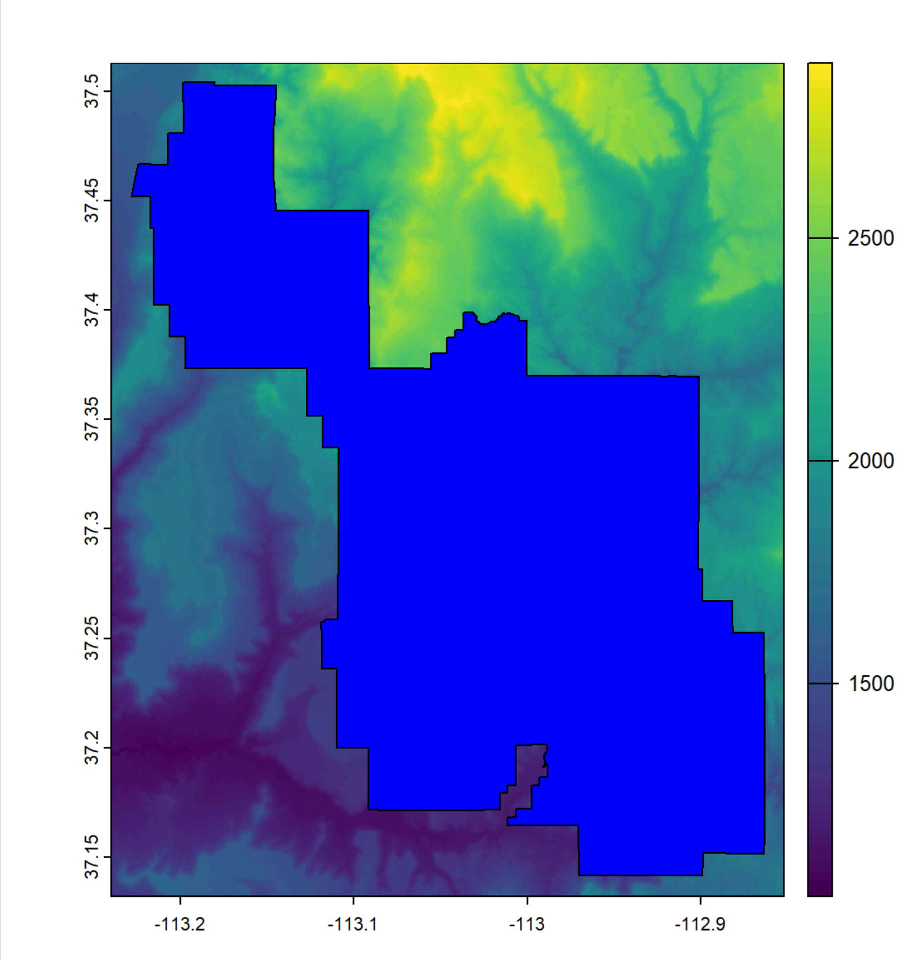
```r
plot(rd)
```



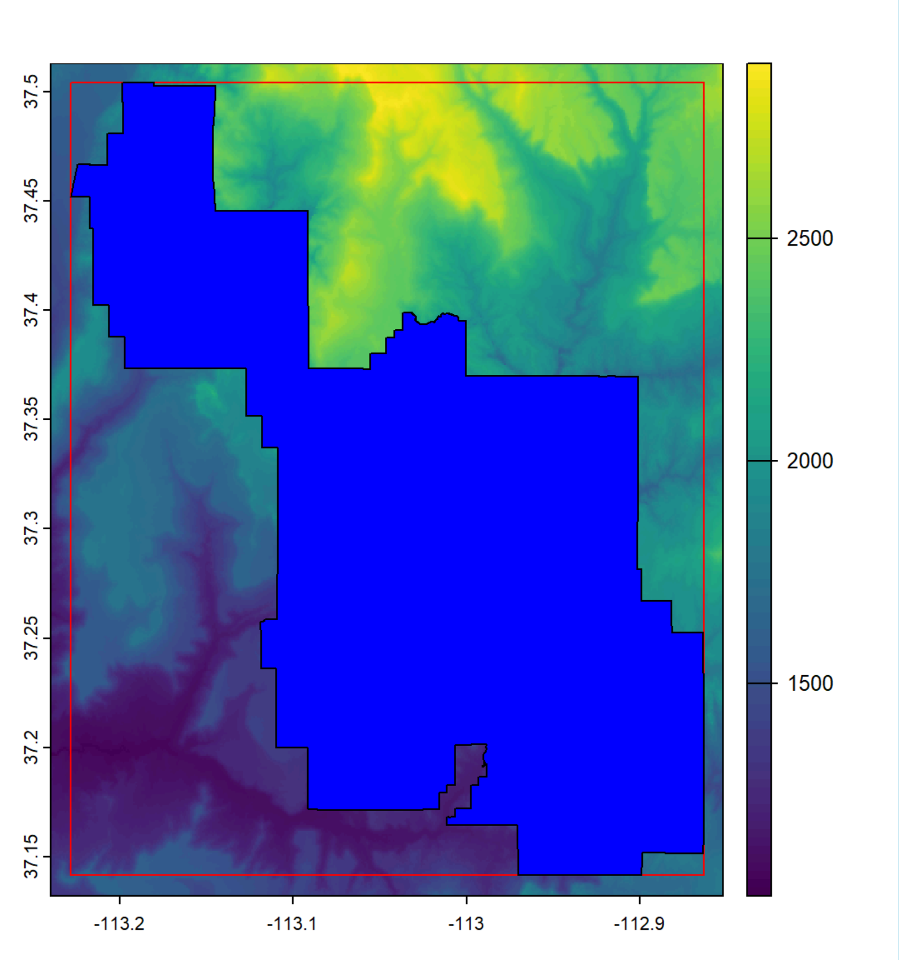
Modifying the Extent

Dealing with Different Extents

- Raster extents often larger than our analysis
- Reducing memory and computational resources
- Making attractive maps



Using `terra::crop()`



- Coordinate Reference System must be the same for both objects
- Crop is based on the (converted) `SpatExtent` of the 2nd object
- `snap` describes how `y` will be aligned to the raster
- Returns all data within the extent

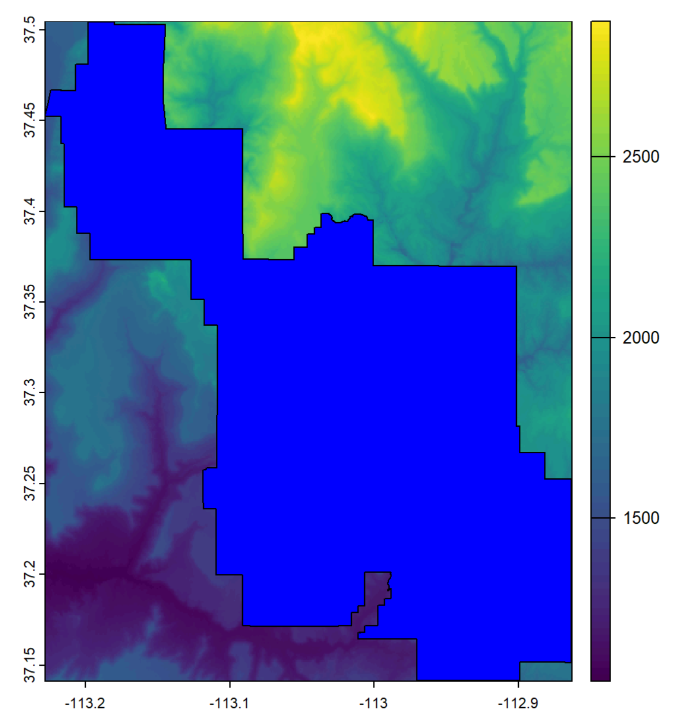
Using `terra::crop()`

```r
library(sf)
library(terra)
library(spDataLarge)
srtm = rast(system.file("raster/srtm.tif", package =
zion = read_sf(system.file("vector/zion.gpkg", packag
zion = st_transform(zion, crs(srtm))

crs(srtm) == crs(zion)
```

```
[1] TRUE
```

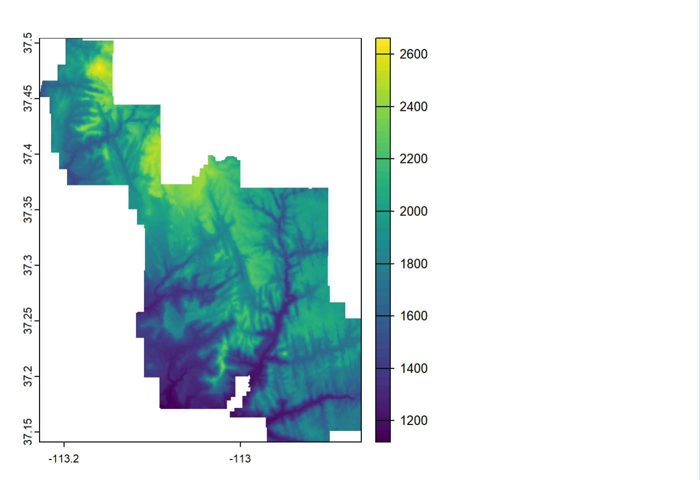
```r
srtm.crop <- crop(x=srtm, y=zion, snap="near")
```



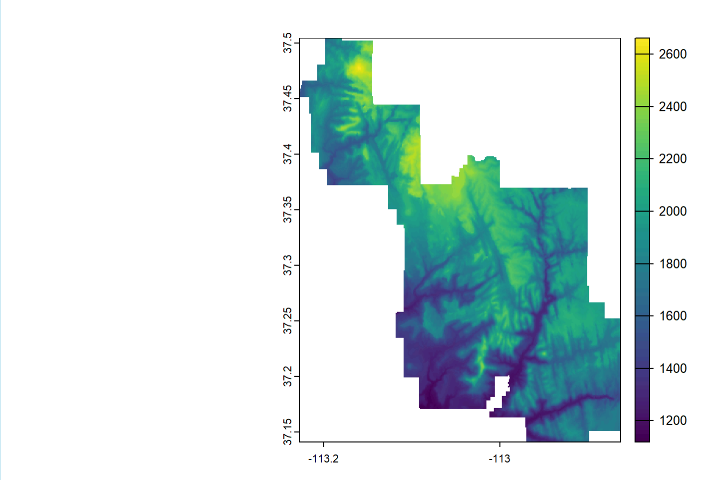
Using `mask()`

- Often want to get rid of all values outside of vector
- Can set `mask=TRUE` in `crop()` (`y` must be `SpatVector`)
- Or use `mask()`

```
srtm.crop.msk <- crop(x=s
plot(srtm.crop.msk)
```



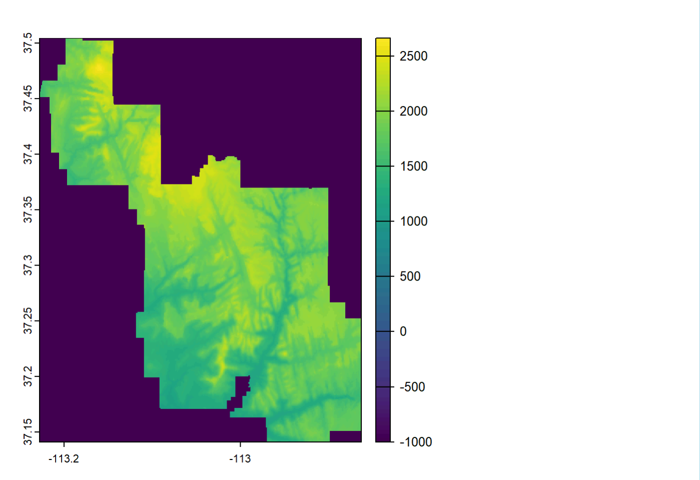
```
srtm.msk <- mask(srtm.cro
plot(srtm.msk)
```



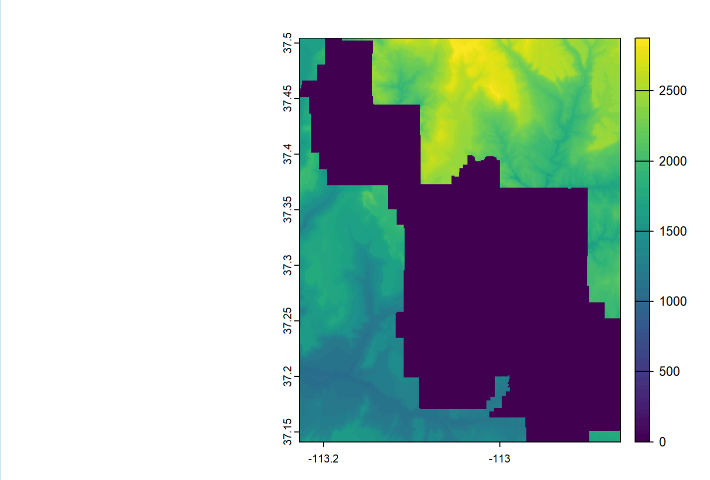
Using `mask()`

- Allows more control over what the mask does
- Can set `maskvalues` and `updatevalues` to change the resulting raster
- Can also use `inverse` to mask out the vector

```r
srtm.msk <- mask(srtm.cro
plot(srtm.msk)
```



```r
srtm.msk <- mask(srtm.cro
plot(srtm.msk)
```



Extending boundaries

- Vector slightly larger than raster
- Especially when using buffered datasets
- Can use `extend`
- Not exact; depends on `snap()`

Practice

`mosaic` is a funciton that combines adjacent rasters, but they need to have the same origin and resolution. Let's practice preparing some rasters for a mosaic.

1. Load wildfire hazard data from the `rasterexample` folder for OR and ID: `Copy of CRPS_OR.tif` and `Copy of CRPS_ID.tif`.
2. These rasters have a fine resolution that will make our calculations slow. Transform them to have a resolution of 900 m.
3. Do the rasters have the same CRS, origin, resolution, and extent? Check this with `==`.
4. Use new functions from this lecture to align the properties mentioned in #3 (plot often to check your work). Why not use `project`?
5. `mosaic` the two rasters together.