Spatial Computations and Rasters

Grayson White

Math 241

Week 5 | Spring 2026

Week 5 Goals

Mon Lecture

- Introduction to spatial data
  - point data
  - polygon data
- Static and interactive maps with point and polygon data
  - base layers (almost as good as merino)
  - coordinate reference systems / projections
  - popups and additional context
  - static graphs with `sf` and `ggplot2`
  - interactive graphs with `leaflet`

Wed Lecture

- Geospatial computations
  - distance
  - spatial joins
- Raster data
  - what is it?
  - rasters as base layers
  - rasters as data
  - extracting values by polygons

Geospatial Computations

Geospatial Computations

Just like we can perform computations with numerical or factor data, we can compute new variables using spatial data

We'll look at a few helpful `sf` functions:

- `st_distance()`: compute distances between spatial objects
- `st_area()`: compute area enclosed by a spatial object
- `st_centroid()`: compute the centroid of a spatial object
- `st_length()`: compute the length of a spatial object
- `st_intersection()` and `st_difference()`: compute intersections or set differences of spatial objects
- `st_join()`: merge spatial data based on spatial relationships

New data: MacLeish Property

- liberal arts field station at Smith College
- 250 acre area of forest and pasture land near Whately, MA
- Smith College community can use the space for artistic inquiry, environmental research, outdoor education, low-impact recreation

Can read in spatial data from `macleish` package

```
1 library(macleish)
2 str(macleish_layers, max.level = 1)
```

```
List of 12
 $ landmarks        :Classes 'sf' and 'data.frame': 14 obs. of  2 variables:
  ..- attr(*, "sf_column")= chr "geometry"
  ..- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA
  .. ..- attr(*, "names")= chr "Label"
 $ forests          :Classes 'sf' and 'data.frame': 54 obs. of  5 variables:
  ..- attr(*, "sf_column")= chr "geometry"
  ..- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA
  .. ..- attr(*, "names")= chr [1:4] "VegType_20" "VegType_21" "Sheet1__Ve" "type"
 $ streams          :Classes 'sf' and 'data.frame': 13 obs. of  2 variables:
  ..- attr(*, "sf_column")= chr "geometry"
  ..- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA
  .. ..- attr(*, "names")= chr "Id"
 $ challenge_courses:Classes 'sf' and 'data.frame': 12 obs. of  24 variables:
```

MacLeish Property

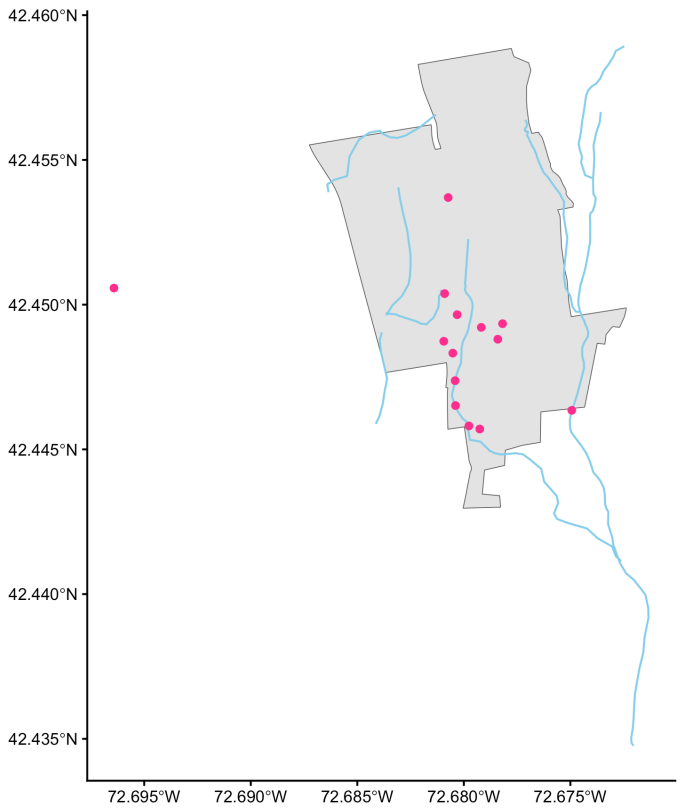
We'll start by taking a look at:

- The `landmarks` on the property (POINTS)
- The `boundary` of the property (POLYGON)
- The `streams` that go through the property (New geometry: LINESTRING)

```
landmarks <- macleish_layers$landmarks
boundary <- macleish_layers$boundary
streams <- macleish_layers$streams
```

MacLeish Property: Plotting

```
ggplot() +
  geom_sf(data = boundary) +
  geom_sf(data = streams, color = "skyblue") +
  geom_sf(data = landmarks, color = "deeppink") +
  theme_classic()
```



Distances: `st_distance()`

Can use the `st_distance()` function to compute the distance between two spatial objects:

```
1 library(sf)
2 # calculate distance between the first five landmarks
3 distance_matrix <- st_distance(landmarks[1:5, ])
4 distance_matrix
```

```
Units: [m]
         [,1]      [,2]      [,3]      [,4]      [,5]
[1,]    0.000 1328.4169 1333.0100 1824.4163 1390.4456
[2,] 1328.417    0.0000  598.0400  508.1877  201.6394
[3,] 1333.010  598.0400    0.0000  946.0184  799.6426
[4,] 1824.416  508.1877  946.0184    0.0000  448.1404
[5,] 1390.446  201.6394  799.6426  448.1404    0.0000
```

```
1 library(units)
2 distance_matrix %>%
3   set_units("mi") # distance in miles
```

```
Units: [mi]
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.0000000 0.8254400 0.8282940 1.1336397 0.8639829
[2,] 0.8254400 0.0000000 0.3716048 0.3157732 0.1252929
[3,] 0.8282940 0.3716048 0.0000000 0.5878286 0.4968749
[4,] 1.1336397 0.3157732 0.5878286 0.0000000 0.2784615
[5,] 0.8639829 0.1252929 0.4968749 0.2784615 0.0000000
```

This distance is **"as the crow flies"**

- There are other options (e.g., distance by driving) – see Chapter 18 in MDSR!

Calculating area: `st_area()`

The `boundary` dataset already reports the area of the plot:

```
1 boundary$area
```

```
255.0989 [acre]
```

But we could calculate it ourselves with the `st_area()` function

```
1 st_area(boundary) %>%
2   set_units("acres")
```

```
255.0989 [acres]
```

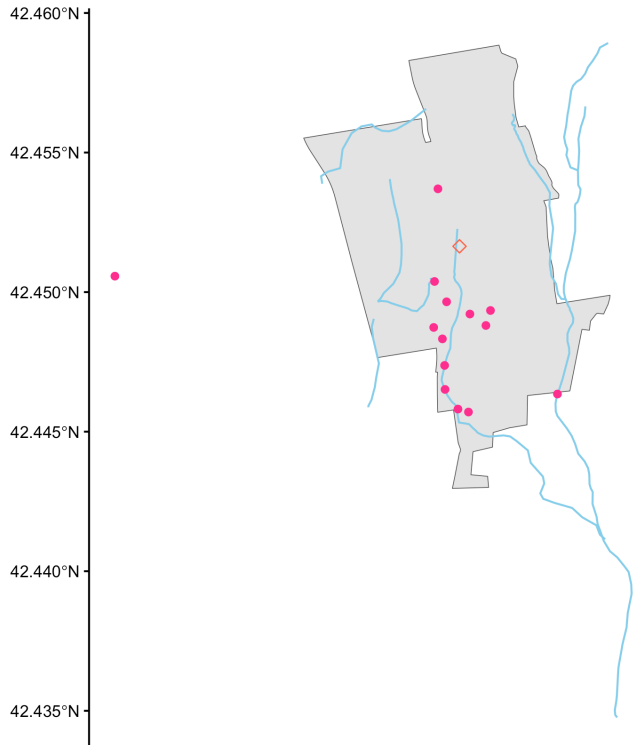
Centroids: `st_centroid()`

Can calculate the centroid of the `boundary` using the `st_centroid()`

```
boundary_center <- st_centroid(boundary)
boundary_center
```

```
Simple feature collection with 1 feature and 1 field
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: -72.67969 ymin: 42.45164 xmax: -72.67969 ymax: 42.45164
Geodetic CRS:  WGS 84
             area                    geometry
1 255.0989 [acre] POINT (-72.67969 42.45164)
```

```
ggplot() +
  geom_sf(data = boundary) +
  geom_sf(data = streams,
          color = "skyblue") +
  geom_sf(data = landmarks,
          color = "deeppink") +
  geom_sf(data = boundary_center,
          shape = 5, size = 2,
          color = "tomato") +
  labs(caption = "Red diamond indicates boundary center
  theme_classic()
```



Lengths of lines/edges: `st_length()`

Can compute the length of each stream with the `st_length()` function (from `sf`)

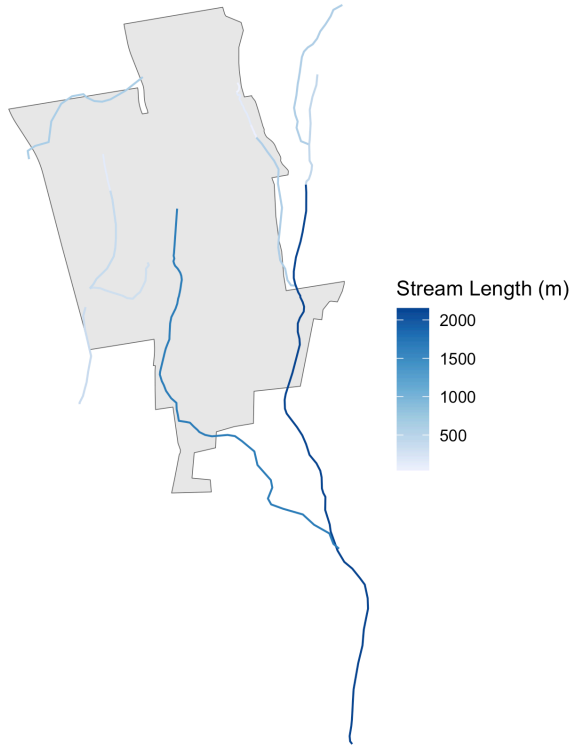
```
1 streams <- streams %>%
2   mutate(length = st_length(geometry))
3 streams
```

```
Simple feature collection with 13 features and 2 fields
Geometry type: LINESTRING
Dimension:     XY
Bounding box:  xmin: -72.68641 ymin: 42.43476 xmax: -72.67133 ymax: 42.45892
Geodetic CRS:  WGS 84
First 10 features:
   Id                       geometry         length
1   1 LINESTRING (-72.67247 42.45...  593.33436 [m]
2   1 LINESTRING (-72.67357 42.45...  412.27916 [m]
3   1 LINESTRING (-72.67652 42.45...  137.87960 [m]
4   1 LINESTRING (-72.67715 42.45...   40.25734 [m]
5   1 LINESTRING (-72.67652 42.45...   51.02365 [m]
6   1 LINESTRING (-72.67628 42.45...  592.79892 [m]
7   1 LINESTRING (-72.67409 42.45... 2152.43892 [m]
```

Lengths of lines/edges: `st_length()`

Coloring streams by length:

```
ggplot(boundary) +
  geom_sf() +
  geom_sf(data = streams, aes(color = as.numeric(length
  scale_color_distiller(palette = "Blues", direction =
  labs(color = "Stream Length (m)") +
  theme_void()
```



Intersections: `st_intersection()` and `st_difference()`

`st_intersection()` can retain only the parts of the streams that lie *within* the boundary:

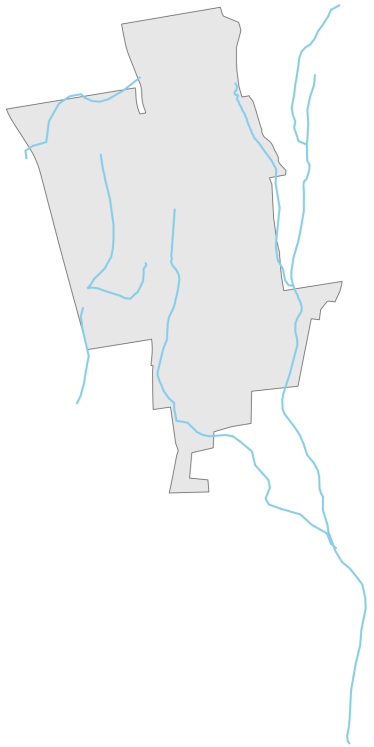
```
1 streams_within <- st_intersection(streams, boundary)
2 streams_within
```

```
Simple feature collection with 11 features and 3 fields
Geometry type: GEOMETRY
Dimension:     XY
Bounding box:  xmin: -72.68622 ymin: 42.44482 xmax: -72.67416 ymax: 42.45657
Geodetic CRS:  WGS 84
First 10 features:
   Id         length            area                        geometry
3   1  137.87960 [m] 255.0989 [acre] LINESTRING (-72.67652 42.45...
4   1   40.25734 [m] 255.0989 [acre] LINESTRING (-72.67715 42.45...
5   1   51.02365 [m] 255.0989 [acre] LINESTRING (-72.67652 42.45...
6   1  592.79892 [m] 255.0989 [acre] MULTILINESTRING ((-72.67628...
7   1 2152.43892 [m] 255.0989 [acre] LINESTRING (-72.67448 42.44...
8   3 1651.25471 [m] 255.0989 [acre] LINESTRING (-72.67979 42.45...
9   3  316.21854 [m] 255.0989 [acre] LINESTRING (-72.68109 42.45...
```

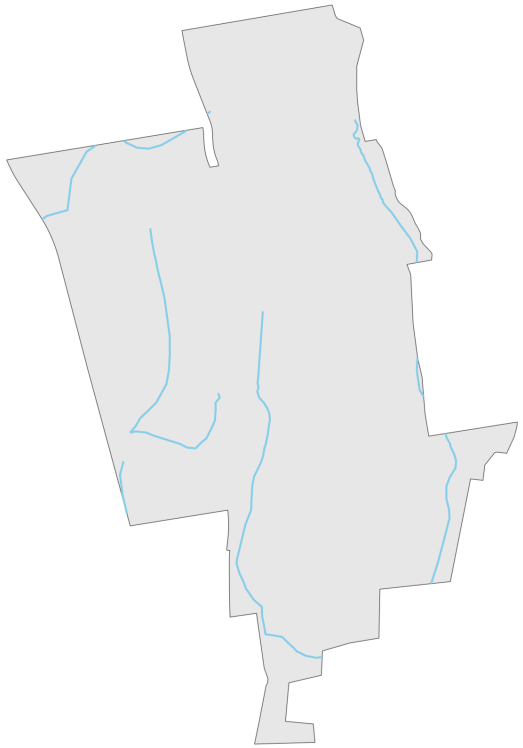
- Note that the intersection returns the **lines** from `streams` not the polygon from `boundary`.

Intersections: `st_intersection()` and `st_difference()`

```
ggplot(boundary) +
  geom_sf() +
  geom_sf(data = streams, color = "skyblue") +
  theme_void()
```



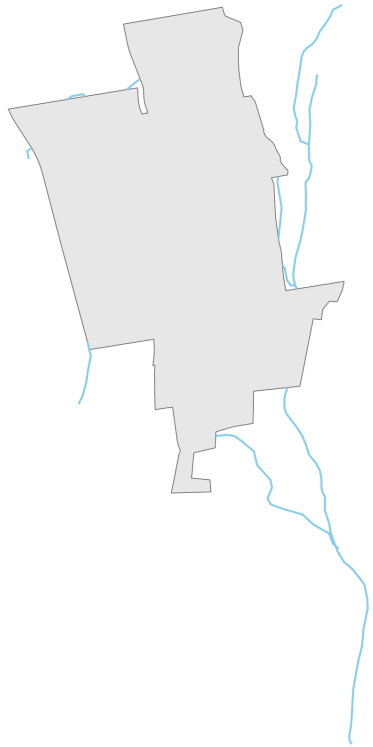
```
ggplot(boundary) +
  geom_sf() +
  geom_sf(data = streams_within, color = "skyblue") +
  theme_void()
```



Intersections: `st_intersection()` and `st_difference()`

`st_difference()` can retain only the parts of the streams that lie *outside* the boundary:

```
streams_outside <- st_difference(streams, boundary)
ggplot(boundary) +
  geom_sf() +
  geom_sf(data = streams_outside, color = "skyblue") +
  theme_void()
```



Intersections: `st_intersection()` and `st_difference()`

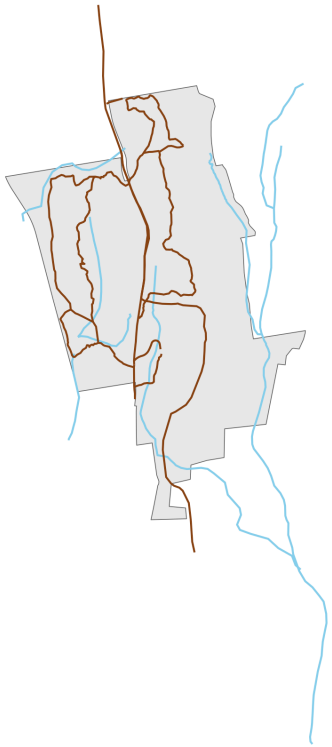
Another example: Let's pull out the trails in this station

```
1 trails <- macleish_layers$trails
2 trails
```

```
Simple feature collection with 15 features and 2 fields
Geometry type: LINESTRING
Dimension:     XY
Bounding box:  xmin: -72.68513 ymin: 42.44177 xmax: -72.67731 ymax: 42.4618
Geodetic CRS:  WGS 84
First 10 features:
               name  color                       geometry
1   Porcupine Trail  White LINESTRING (-72.68291 42.45...
2      Western Loop    Red LINESTRING (-72.68111 42.45...
3  Poplar Hill Road   Road LINESTRING (-72.68155 42.45...
4  Vernal Pool Loop Yellow LINESTRING (-72.68265 42.44...
5      Eastern Loop   Blue LINESTRING (-72.68113 42.45...
6      Western Loop    Red LINESTRING (-72.68401 42.45...
7      Western Loop    Red LINESTRING (-72.68088 42.44...
```

Intersections: `st_intersection()` and `st_difference()`

```
ggplot(boundary) +
  geom_sf() +
  geom_sf(data = streams, color = "skyblue") +
  geom_sf(data = trails, color = "chocolate4") +
  theme_void()
```



Intersections: `st_intersection()` and `st_difference()`

…and see where they intersect with the streams!

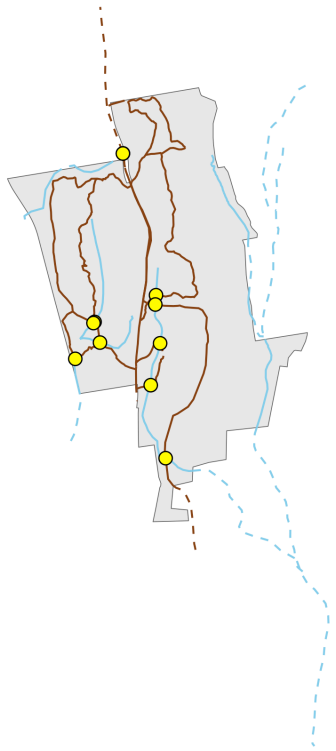
```
1 intersections <- st_intersection(trails, streams)
2 intersections
```

```
Simple feature collection with 10 features and 4 fields
Geometry type: GEOMETRY
Dimension:     XY
Bounding box:  xmin: -72.68389 ymin: 42.44529 xmax: -72.67941 ymax: 42.45644
Geodetic CRS:  WGS 84
                 name  color Id        length                        geometry
8        entry trail      -  3 1651.2547 [m]     POINT (-72.68015 42.44796)
9       Eastern Loop   Blue  3 1651.2547 [m]     POINT (-72.67989 42.45126)
13  Snowmobile Trail   <NA>  3 1651.2547 [m] MULTIPOINT ((-72.67992 42.4...
15          Driveway   <NA>  3 1651.2547 [m]      POINT (-72.67968 42.4495)
6       Western Loop    Red  3  316.2185 [m]     POINT (-72.68267 42.44952)
1     Porcupine Trail  White  3  388.2598 [m]     POINT (-72.68293 42.45028)
6.1     Western Loop    Red  3  388.2598 [m]     POINT (-72.68299 42.45024)
4   Vernal Pool Loop Yellow  3  362.6555 [m]     POINT (-72.68389 42.44893)
```

- Note that **the intersection geometries are now *points***

Intersections: `st_intersection()` and `st_difference()`

```
trails_within <- st_intersection(trails, boundary)
trails_outside <- st_difference(trails, boundary)

ggplot(boundary) +
  geom_sf() +
  geom_sf(data = streams_within,
          color = "skyblue") +
  geom_sf(data = streams_outside,
          color = "skyblue",
          linetype = "dashed") +
  geom_sf(data = trails_within,
          color = "chocolate4") +
  geom_sf(data = trails_outside,
          color = "chocolate4",
          linetype = "dashed") +
  geom_sf(data = intersections,
          shape = 21,
          fill = "yellow",
```



- Plotted stream crossing!

Aggregation

Let's look at the trails data frame a little more:

```
1 trails %>%
2   arrange(name)
```

```
Simple feature collection with 15 features and 2 fields
Geometry type: LINESTRING
Dimension:     XY
Bounding box:  xmin: -72.68513 ymin: 42.44177 xmax: -72.67731 ymax: 42.4618
Geodetic CRS:  WGS 84
First 10 features:
               name color                          geometry
1          Driveway  <NA> LINESTRING (-72.68088 42.44...
2      Eastern Loop  Blue LINESTRING (-72.68113 42.45...
3      Eastern Loop  Blue LINESTRING (-72.68063 42.45...
4          Easy Out   Red LINESTRING (-72.67962 42.45...
5          Easy Out  <NA> LINESTRING (-72.68144 42.45...
6  Poplar Hill Road  Road LINESTRING (-72.68155 42.45...
7  Poplar Hill Road  Road LINESTRING (-72.68087 42.44...
```

What is odd about this data frame?

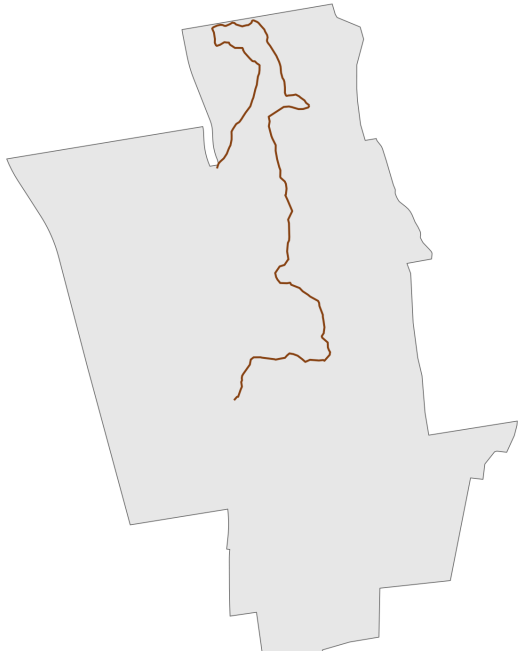
Aggregation

Let's look at the "Eastern Loop"

```
1 eastern <- trails %>% filter(name=="Eastern Loop")
2 eastern
```

```
Simple feature collection with 2 features and 2 fields
Geometry type: LINESTRING
Dimension:     XY
Bounding box:  xmin: -72.68127 ymin: 42.45033 xmax: -72.67783 ymax: 42.4585
Geodetic CRS:  WGS 84
          name color                       geometry
1 Eastern Loop  Blue LINESTRING (-72.68113 42.45...
2 Eastern Loop  Blue LINESTRING (-72.68063 42.45...
```

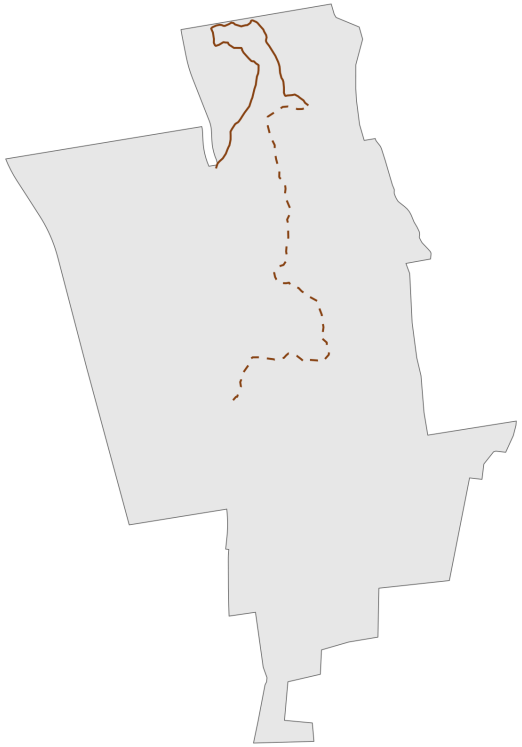
```
1 ggplot(boundary) +
2   geom_sf() +
3   geom_sf(data = eastern, color = "chocolate4") +
4   theme_void()
```



Aggregation

What are the pieces of this trail?

```
part_one <- eastern[1, ]
part_two <- eastern[2, ]
ggplot(boundary) +
  geom_sf() +
  geom_sf(data = part_one,
          color = "chocolate4") + # part 1
  geom_sf(data = part_two,
          color = "chocolate4",
          linetype = "dashed") + # part 2
  theme_void()
```



Aggregation

For plotting this is fine, but what if we wanted to compare the lengths of each trail?

```
1 trails %>%
2   mutate(trail_length = st_length(geometry)) %>%
3   arrange(desc(trail_length))
```

```
Simple feature collection with 15 features and 3 fields
Geometry type: LINESTRING
Dimension:     XY
Bounding box:  xmin: -72.68513 ymin: 42.44177 xmax: -72.67731 ymax: 42.4618
Geodetic CRS:  WGS 84
First 10 features:
               name  color  trail_length                       geometry
1  Snowmobile Trail   <NA> 1346.9147 [m] LINESTRING (-72.68053 42.45...
2  Snowmobile Trail   <NA> 1229.1787 [m] LINESTRING (-72.68053 42.45...
3      Eastern Loop   Blue 1107.9588 [m] LINESTRING (-72.68063 42.45...
4      Western Loop    Red  969.0932 [m] LINESTRING (-72.68111 42.45...
5  Poplar Hill Road   Road  900.5781 [m] LINESTRING (-72.68155 42.45...
6      Eastern Loop   Blue  831.2864 [m] LINESTRING (-72.68113 42.45...
7   Porcupine Trail  White  699.7393 [m] LINESTRING (-72.68291 42.45...
```

The same trail comes up multiple times! How can we combine them?

- `group_by()` and `summarize()`

Aggregation: `group_by` and `summarize()`

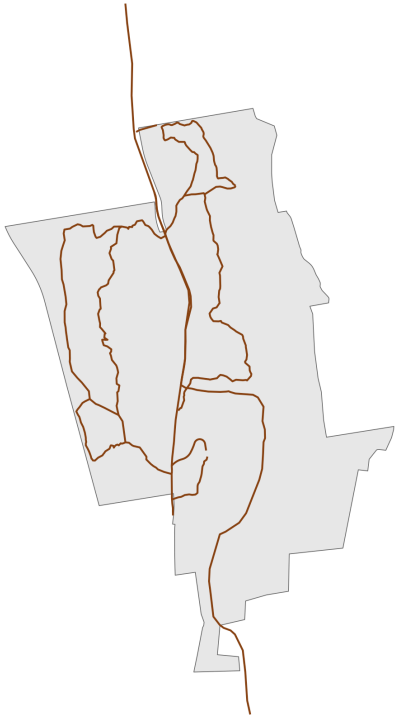
```
1 trails_combined <- trails %>%
2   group_by(name) %>%
3   summarize(trail_length = sum(st_length(geometry))) %>%
4   arrange(desc(trail_length))
5 trails_combined
```

```
Simple feature collection with 9 features and 2 fields
Geometry type: GEOMETRY
Dimension:     XY
Bounding box:  xmin: -72.68513 ymin: 42.44177 xmax: -72.67731 ymax: 42.4618
Geodetic CRS:  WGS 84
# A tibble: 9 × 3
  name             trail_length                                    geometry
  <chr>                   [m]                                <GEOMETRY [°]>
1 Snowmobile Trail        2576. MULTILINESTRING ((-72.68053 42.45106, -72.68035…
2 Eastern Loop            1939. MULTILINESTRING ((-72.68113 42.45531, -72.68111…
3 Western Loop            1350. MULTILINESTRING ((-72.68111 42.45526, -72.68129…
4 Poplar Hill Road        1040. MULTILINESTRING ((-72.68155 42.45649, -72.68149…
5 Porcupine Trail          700. LINESTRING (-72.68291 42.45021, -72.68296 42.45…
6 Vernal Pool Loop         360. LINESTRING (-72.68265 42.44944, -72.68274 42.44…
```

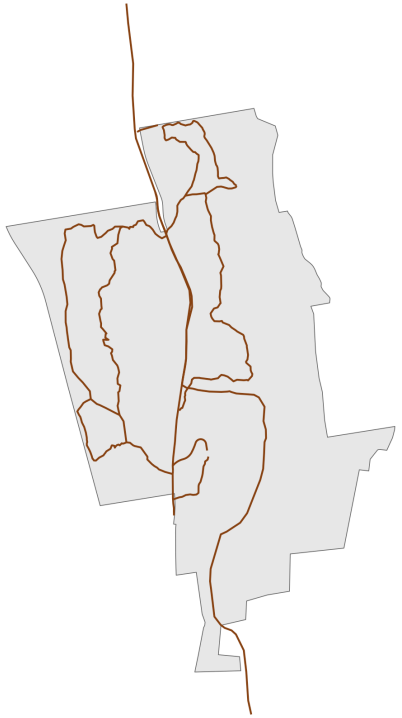
Only one row now for each trail.

Aggregation: `group_by` and `summarize()`

```
# Not combining
ggplot(boundary) +
  geom_sf() +
  geom_sf(data = trails,
          color = "chocolate4") +
  theme_void()
```



```
# Combined
ggplot(boundary) +
  geom_sf() +
  geom_sf(data = trails_combined,
          color = "chocolate4") +
  theme_void()
```



01:00

Find the difference in the plots!!

Aggregation: `group_by` and `summarize()`

Merging Spatial data: `st_join()`

```
forests <- macleish_layers$forests %>%
  select(type, geometry)
camp_sites <- macleish_layers$camp_sites %>%
  select(name, geometry)
```

```
forests
```

```
Simple feature collection with 54 features and 1 field
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -72.68726 ymin: 42.44292 xmax:
-72.67237 ymax: 42.45884
Geodetic CRS:  WGS 84
First 10 features:
                            type
geometry
1                           <NA> MULTIPOLYGON (((-72.68079
4...
2                Abandoned Pasture MULTIPOLYGON (((-72.67814
4...
3     Active Pasture and Hayfield MULTIPOLYGON (((-72.67974
```

```
camp_sites
```

```
Simple feature collection with 2 features and 1 field
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: -72.67958 ymin: 42.45098 xmax:
-72.67815 ymax: 42.45855
Geodetic CRS:  WGS 84
# A tibble: 2 × 2
  name                        geometry
  <chr>                    <POINT [°]>
1 Group Campsite  (-72.67815 42.45098)
2 Remote Campsite (-72.67958 42.45855)
```

Merging Spatial data: `st_join()`

```
library(ggsflabel)
ggplot(boundary) +
  geom_sf() +
  geom_sf(data = forests, fill = "forestgreen",
          alpha = 0.2) +
  geom_sf(data = camp_sites, size = 4) +
  geom_sf_label_repel(data = camp_sites,
                      aes(label = name)) +
  theme_void()
```

What if we wanted to know **which forests** these campsites are in?

Merging Spatial data: `st_join()`

`st_join()` can merge datasets on spatial data.

```
1 camp_sites_forests <- st_join(camp_sites, forests, left = TRUE)
2 camp_sites_forests
```

```
Simple feature collection with 2 features and 2 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: -72.67958 ymin: 42.45098 xmax: -72.67815 ymax: 42.45855
Geodetic CRS:  WGS 84
# A tibble: 2 × 3
  name                       geometry type
* <chr>                   <POINT [°]> <chr>
1 Group Campsite  (-72.67815 42.45098) Old Field White Pine Forest
2 Remote Campsite (-72.67958 42.45855) Sugar Maple Forest
```

- Setting `left = TRUE` yields a `left_join()` here. `left = FALSE` would yield an `inner_join()`.

You can set the `join` argument to specify *how* you want to connect the geometries.

- `join = st_interests` is the default, but other `st_*()` functions can be specified.

Raster data

What are raster data?

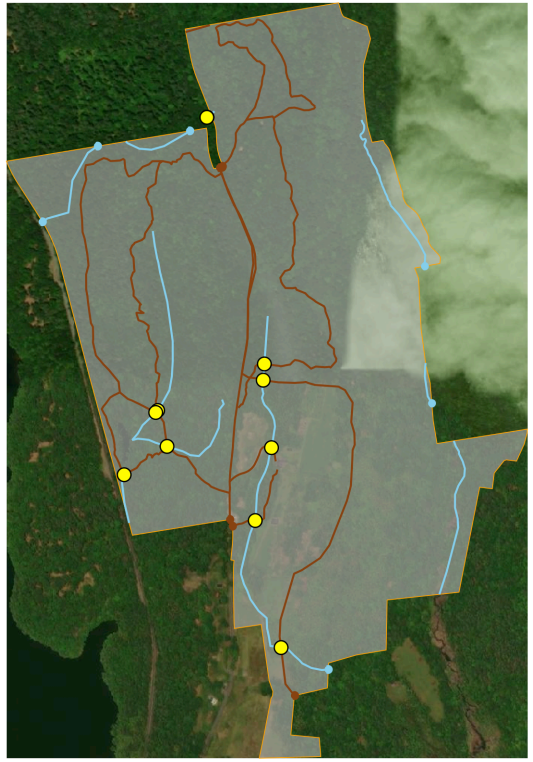
- So far, we've dealt with (multi)points, (multi)polygons, and line(string) data!
  - These all fall into the family of "vector" spatial data
  - Called "vectors" since they are made up of discrete x, y coordinates that compose a shape.
- Now, we are entering into the world of **raster** spatial data.
- Raster data are arrays of spatially indexed cells, each of the same size
- Think: **spatially indexed matrix** or **image**. Each cell / pixel..
  - is a predefined size square
  - is the same size as the other cells / pixels
  - has value(s) attached to it

Raster data: two hats

- We'll use raster data in two distinct ways:
  - as a **base layer** for maps, and
  - as **data** for maps.

Raster as base layer: example

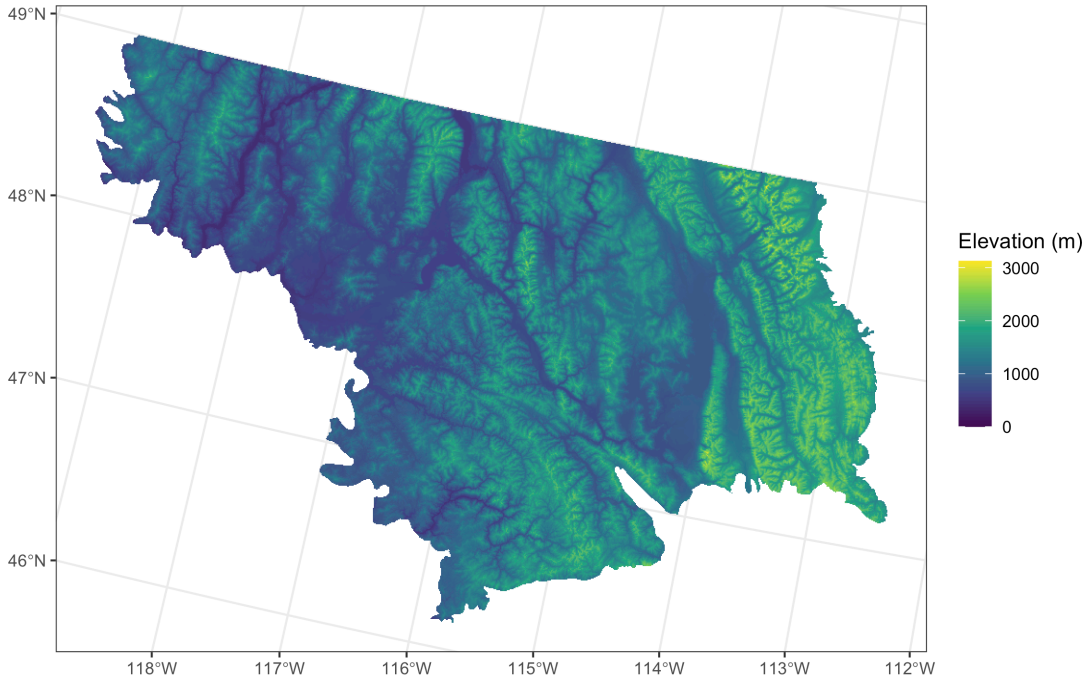
2.39m x 2.39m imagery in the Macleish property



- Each pixel is a 2.39m x 2.39m square and has RED, GREEN, BLUE (RGB) values assigned to it.

Raster as data: example

90m x 90m elevation data in M333 “ecoprovince” (NE Washington, N Idaho, W Montana)



- Each pixel is a 90m x 90m square and has an elevation value assigned to it.

Loading raster data

We'll use the `terra` R package to deal with rasters.

- `terra` can also handle many vector data types such as points and polygons.
  - But `sf` is often nicer to work with for those data types.
- Raster data are often stored as Geotiff (.tif) files.

```
1 library(terra)
2 r <- terra::rast("data/M333_elev.tif")
3 class(r)
```

```
[1] "SpatRaster"
attr(,"package")
[1] "terra"
```

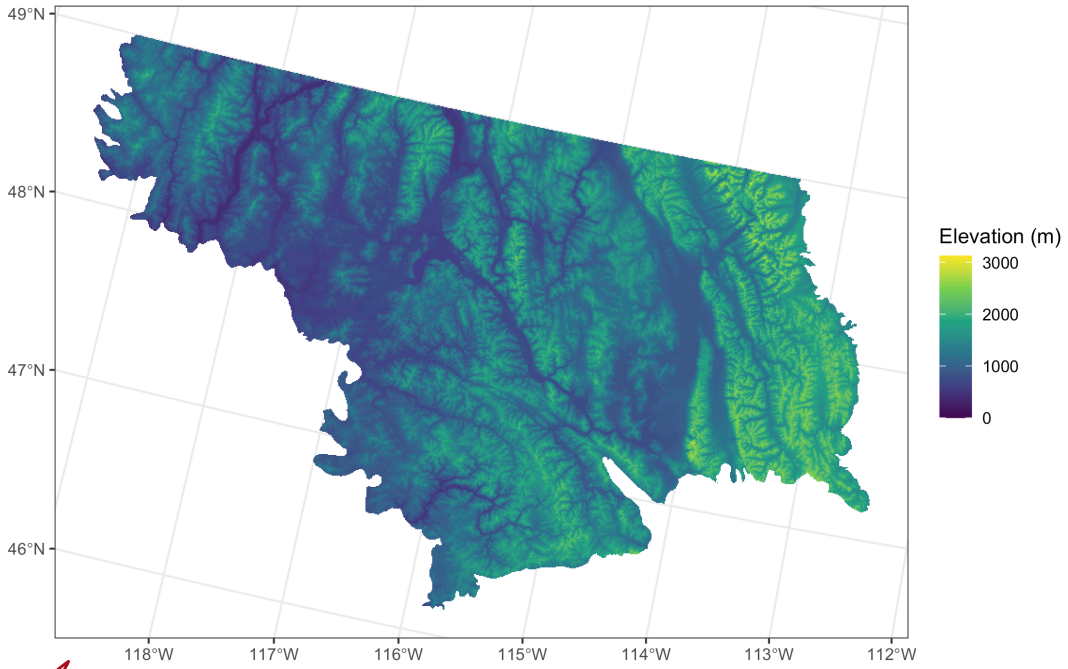
```
1 r
```

```
class       : SpatRaster
size        : 4083, 5509, 1  (nrow, ncol, nlyr)
resolution  : 90, 90  (x, y)
extent      : -1736295, -1240485, 2731305, 3098775  (xmin, xmax, ymin, ymax)
coord. ref. : NAD_1983_Albers
source      : M333_elev.tif
name        : M333_elev
```

Plotting raster data

The `tidyterra` package gives us some nice `geom_*()` functions to use with `SpatRaster`s and `SpatVector`s.

```
1 library(tidyterra)
2 ggplot() +
3   geom_spatraster(data = r, na.rm = TRUE) +
4   scale_fill_viridis_c(na.value = NA) +
5   labs(fill = "Elevation (m)") +
6   theme_bw()
```



- Main functions:
  - `geom_spatraster()` for rasters as data
  - `geom_spatraster_rgb()` for rasters as baselayers

Raster attributes

Rasters have CRSs as well!

```
1 crs(r)
```

```
[1] "PROJCRS[\"NAD_1983_Albers\",\n    BASEGEOGCRS[\"NAD83\",\n        DATUM[\"North American Datum 1983\",\n
ELLIPSOID[\"GRS 1980\",6378137,298.257222101004,\n                LENGTHUNIT[\"metre\",1]]],\n
PRIMEM[\"Greenwich\",0,\n            ANGLEUNIT[\"degree\",0.0174532925199433]],\n        ID[\"EPSG\",4269]],\n
CONVERSION[\"Albers Equal Area\",\n        METHOD[\"Albers Equal Area\",\n            ID[\"EPSG\",9822]],\n
PARAMETER[\"Latitude of false origin\",23,\n            ANGLEUNIT[\"degree\",0.0174532925199433],\n
ID[\"EPSG\",8821]],\n        PARAMETER[\"Longitude of false origin\",-96,\n
ANGLEUNIT[\"degree\",0.0174532925199433],\n            ID[\"EPSG\",8822]],\n        PARAMETER[\"Latitude of 1st standard
parallel\",29.5,\n            ANGLEUNIT[\"degree\",0.0174532925199433],\n            ID[\"EPSG\",8823]],\n
PARAMETER[\"Latitude of 2nd standard parallel\",45.5,\n            ANGLEUNIT[\"degree\",0.0174532925199433],\n
ID[\"EPSG\",8824]],\n        PARAMETER[\"Easting at false origin\",0,\n            LENGTHUNIT[\"metre\",1],\n
ID[\"EPSG\",8826]],\n        PARAMETER[\"Northing at false origin\",0,\n            LENGTHUNIT[\"metre\",1],\n
ID[\"EPSG\",8827]]],\n    CS[Cartesian,2],\n        AXIS[\"easting\",east,\n            ORDER[1],\n
LENGTHUNIT[\"metre\",1,\n                ID[\"EPSG\",9001]]],\n        AXIS[\"northing\",north,\n            ORDER[2],\n
LENGTHUNIT[\"metre\",1,\n                ID[\"EPSG\",9001]]]]"
```

And we can get summary stats from rasters

```
1 max(values(r), na.rm = TRUE) # highest elevation in M333
```

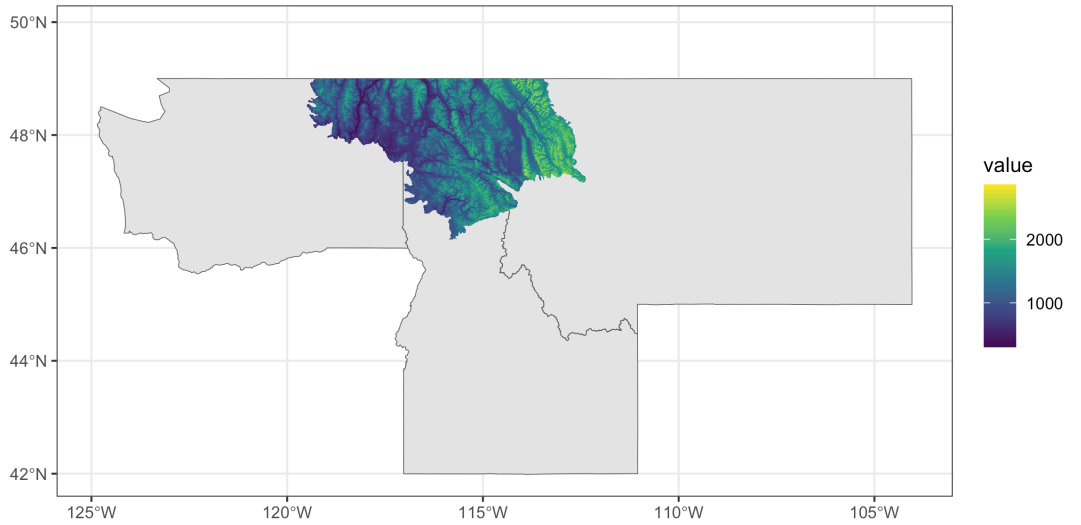
```
[1] 3188
```

```
1 mean(values(r), na.rm = TRUE) # average elevation in M333
```

```
[1] 1256.759
```

Plotting rasters with vector data

```
library(tigris)
states(progress_bar = FALSE) %>%
  filter(NAME %in% c("Washington",
                     "Idaho",
                     "Montana")) %>%
  ggplot() +
  geom_sf() +
  geom_spatraster(data = r) +
  scale_fill_viridis_c(na.value = NA) +
  theme_bw()
```



- Since both the raster and polygons are spatially referenced they “play nice” together!

Extracting values from vectors that intersect our raster

- Sometimes, we'd like to know the value of a raster a specific location
  - Think: elevation at a given point, or
  - average elevation in a national forest, etc.
- `terra` makes this easy.
- Let's get the elevation of the FIA plots in the `forested` package that intersect with our raster!

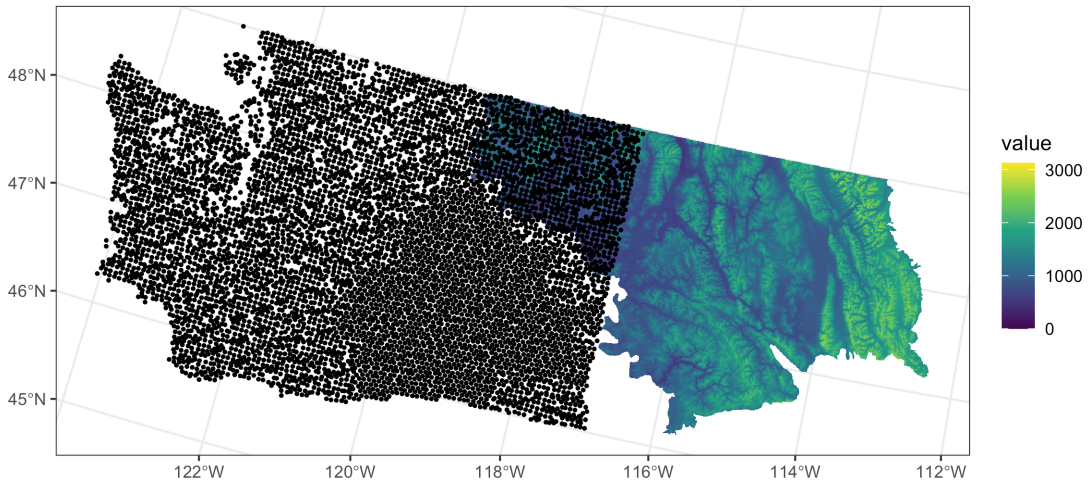
```
1 library(forested)
2 forested_sf <- forested %>%
3   st_as_sf(coords = c("lon", "lat"),
4            crs = "EPSG:4269")
5
6 plot_elev <- terra::extract(r, forested_sf, xy = TRUE)
7 plot_elev
```

```
      ID M333_elev        x       y
1      1       881 -1672800 3055350
2      2        NA      NaN     NaN
3      3        NA      NaN     NaN
4      4        NA      NaN     NaN
5      5       806 -1630320 2974890
6      6       736 -1615560 2984610
7      7       636 -1575240 2986500
8      8        NA      NaN     NaN
9      9        NA      NaN     NaN
10    10        NA      NaN     NaN
11    11        NA      NaN     NaN
12    12        NA      NaN     NaN
13    13        NA      NaN     NaN
```

- Why so many NAs?

Not all plots intersect our raster!

```
ggplot() +
  geom_spatraster(data = r) +
  geom_sf(data = forested_sf, size = 0.5) +
  scale_fill_viridis_c(na.value = NA) +
  theme_bw()
```

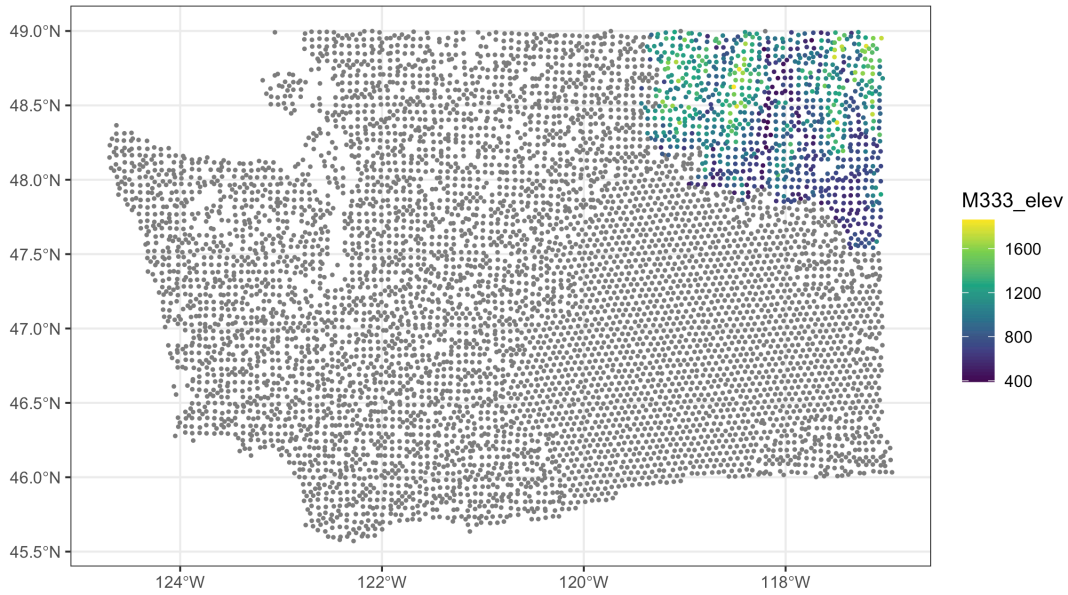


Bringing elevation values into the `forested` data

```
forested_with_M333_elev <- forested_sf %>%
  bind_cols(plot_elev)
```

- `bind_cols()` takes two data frames of the same number of rows and "pushes them together".

```
forested_with_M333_elev %>%
  ggplot() +
  geom_sf(aes(color = M333_elev), size = 0.5) +
  scale_color_viridis_c() +
  theme_bw()
```



Elevation actually already exists in the `forested` dataset

```
mean(forested_with_M333_elev$elevation == forested_with_M333_elev$M333_elev)
```

```
[1] NA
```

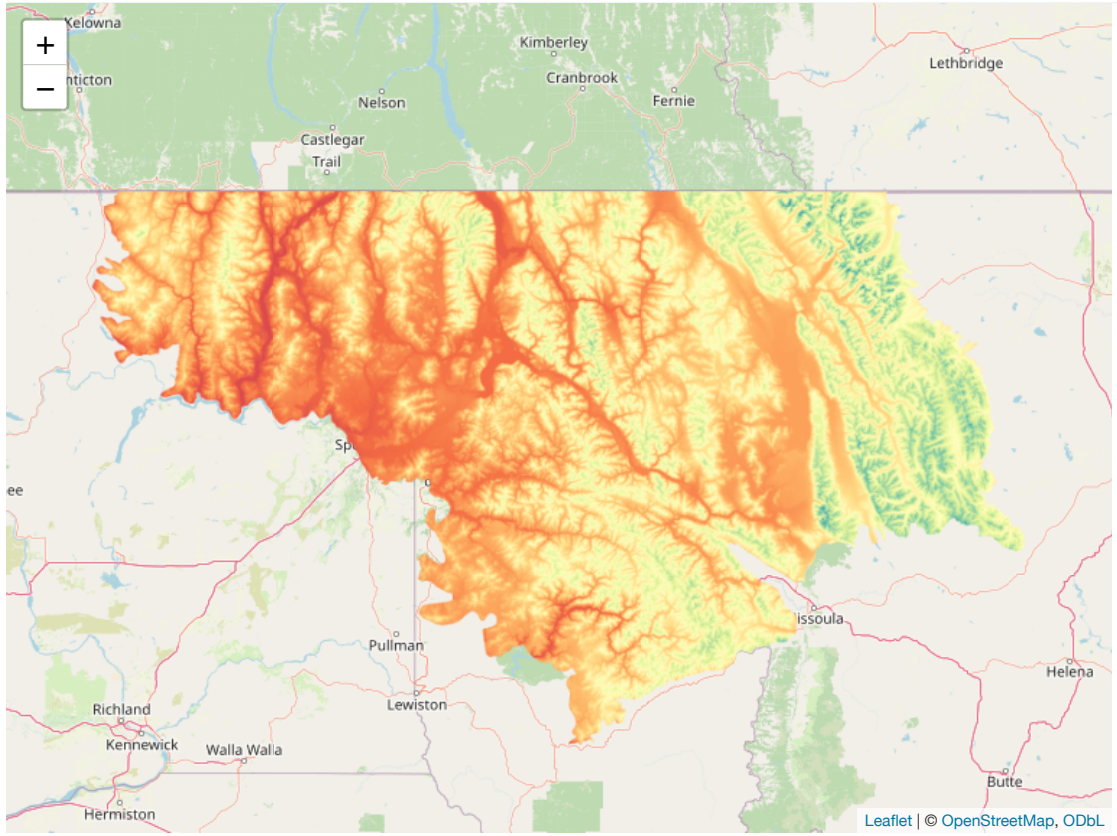
```
mean(forested_with_M333_elev$elevation == forested_with_M333_elev$M333_elev,
     na.rm = TRUE)
```

```
[1] 1
```

- When the data are available, we get the same result as the package!

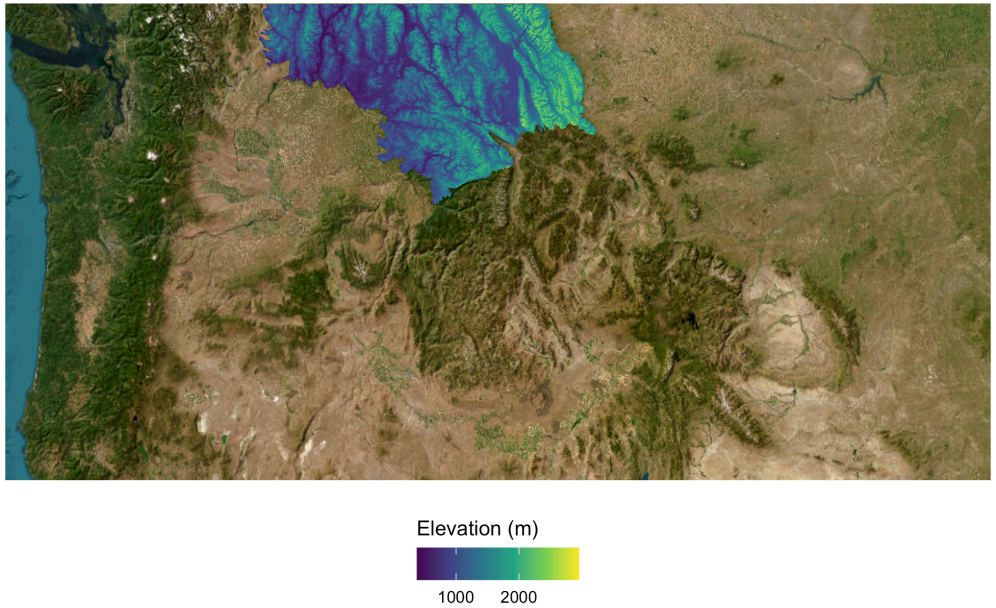
Interactive raster plots

```
1 library(leaflet)
2 leaflet() %>%
3   addTiles() %>%
4   addRasterImage(r, maxBytes = 15000000)
```



Plotting rasters with raster base layers

```
library(basemaps) # to get the raster base layers

wa_id_mt <- tigris::states(progress_bar = FALSE) %>%
  filter(NAME %in% c("Washington",
                     "Idaho",
                     "Montana"))

rast <- basemap_terra(ext = wa_id_mt,
                      map_service = "esri",
                      map_type = "world_imagery",
                      verbose = FALSE)

ggplot() +
  geom_spatraster_rgb(data = rast) +
  geom_spatraster(data = r) +
  scale_fill_viridis_c(na.value = NA) +
  labs(fill = "Elevation (m)") +
  theme_void() +
```

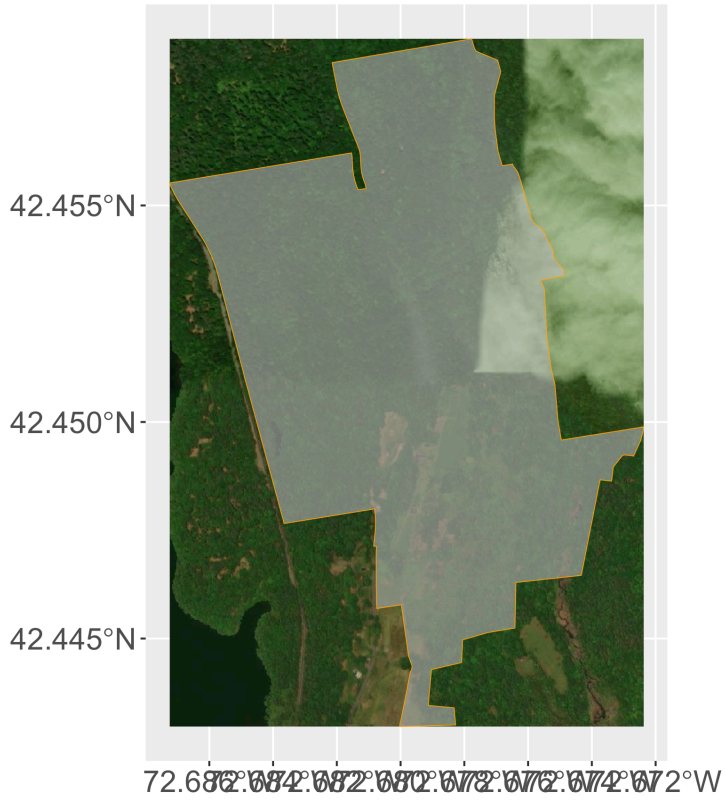


- Using rasters in two ways: base layer and as data!

Returning to the Macleish Property

We can use a raster base layer and put vector data on top of it!

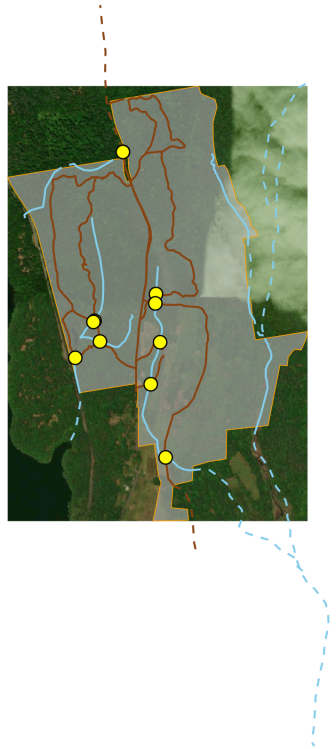
```
rast <- basemap_terra(
  # recall, `boundary` is the boundary
  # of the Macleish property
  ext = boundary,
  map_service = "esri",
  map_type = "world_imagery",
  verbose = FALSE
  )

ggplot() +
  geom_spatraster_rgb(data = rast) +
  geom_sf(data = boundary,
          alpha = 0.5,
          fill = "grey",
          color = "orange")
```



Returning to the Macleish Property

We can use a raster base layer and put vector data on top of it!

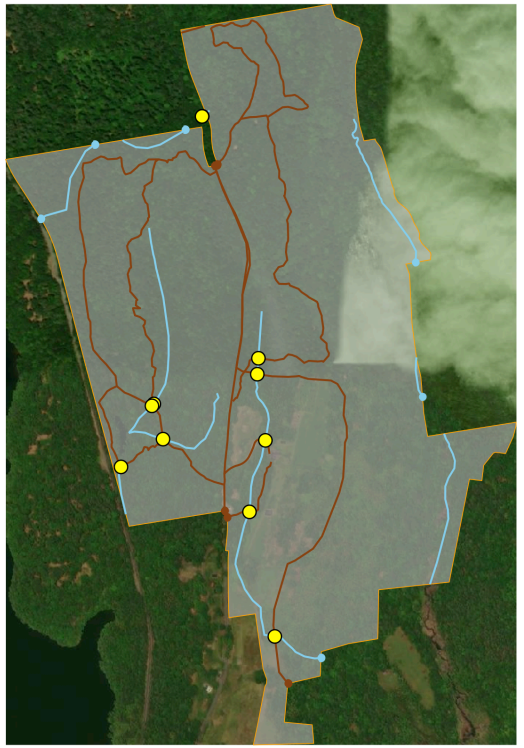
```
ggplot() +
  geom_spatraster_rgb(data = rast) +
  geom_sf(data = boundary,
          alpha = 0.5,
          fill = "grey",
          color = "orange") +
  geom_sf(data = streams_within,
          color = "skyblue") +
  geom_sf(data = streams_outside,
          color = "skyblue",
          linetype = "dashed") +
  geom_sf(data = trails_within,
          color = "chocolate4") +
  geom_sf(data = trails_outside ,
          color = "chocolate4",
          linetype = "dashed") +
  geom_sf(data = intersections,
          shape = 21,
```



Returning to the Macleish Property

We can use a raster base layer and put vector data on top of it!

```
ggplot() +
  geom_spatraster_rgb(data = rast) +
  geom_sf(data = boundary,
          alpha = 0.5,
          fill = "grey",
          color = "orange") +
  geom_sf(data = streams_within,
          color = "skyblue") +
  geom_sf(data = streams_outside %>%
            st_intersection(boundary),
          color = "skyblue",
          linetype = "dashed") +
  geom_sf(data = trails_within,
          color = "chocolate4") +
  geom_sf(data = trails_outside %>%
            st_intersection(boundary),
          color = "chocolate4",
          linetype = "dashed") +
```



- Note the use of `st_intersection()`

Spatial data: just scratching the surface

- We could spend all semester on spatial data…
- There is so much more we can do with rasters
  - raster algebra (adding / multiplying / etc. rasters)
  - reprojecting rasters to different resolutions
- You'll have opportunities in your projects to use spatial data if you'd like to!
- And lots of practice in Problem Set 4!

Next week

- Learn how to create interactive dashboards with `shiny`! (a great way to disseminate all of these interactive plots we've learned about)
- Get Project 1 instructions