Lab 5 Oregon Fires

Lauren Ponisio

Conservation/ecology Topics

- Explore how Oregon fires are changing due to fire suppression and climate change.
- Describe fundamental concepts in fire ecology, including fire severity.

Statistical Topics

- Describe the fundamental attributes of a raster dataset.

Computational Topics

- Explore raster attributes and metadata using R.
- Import rasters into R using the `terra` package.
- Plot raster files in R using the `ggplot2` package.
- Reproject raster and vector data
- Layer raster and vector data together

Lab part 1: reading in fire raster data and plotting

We will be working with the soil burn severity data from the 2020 Holiday Farm Fire (up the McKenzie E of Eugene), the 2020 Beachie Fire (near Portland) and the 2018 Terwilliger fire (up the McKenzie E of Eugene, near Cougar hotsprings).

We will use data downloaded from the USGS: https://burnseverity.cr.usgs.gov/products/baer

Specifically, BARC Fire Severity layers are created by first calculating spectral indices from pre- and post-fire satellite imagery that are sensitive to changes caused by fire. The two images are then subtracted showing the difference between them which is then binned into 4 burn severity classes (high, moderate, low, very low/unburned). Field crews ground-truth the severity classes.

The metadata files provide additional details on how the continuous data was binned into discrete catagories.

a. Read in each fire severity rasters, name them [fire name]_rast. The .tif files are the rasters.

HINT: The files are nested within folders so be aware of your file paths.

```
# DSM_HARV <-
  # rast("data/NEON-DS-Airborne-Remote-Sensing/HARV/DSM/HARV_dsmCrop.tif")

rebel_rast <- rast("soil-burn-severity/2017_rebel_sbs/rebel_sbs2.tif")
```

```
terwilliger_rast <- rast("soil-burn-severity/2018_terwilliger_sbs/SoilSeverity.tif")
beachie_rast <- rast("soil-burn-severity/2020_beachiecreek_sbs/BeachieCreek_SBS_final.tif")
holiday_rast <- rast("soil-burn-severity/2020_holidayfarm_sbs/HolidayFarm_SBS_final.tif")
```

b. Summarize the values of the rasters. Take note of the labels associated with the data values because you will need it for plotting.

```
summary(values(rebel_rast))
```

```
##     Layer_1
##  Min.   : 1.00
##  1st Qu.:15.00
##  Median :15.00
##  Mean   :14.47
##  3rd Qu.:15.00
##  Max.   :15.00
```

```
summary(values(terwilliger_rast))
```

```
##    SoilBurnSe
##  Min.   :1.00
##  1st Qu.:2.00
##  Median :2.00
##  Mean   :1.92
##  3rd Qu.:2.00
##  Max.   :4.00
##  NA's   :80287
```

```
summary(values(beachie_rast))
```

```
##     Layer_1
##  Min.   :  1.00
##  1st Qu.:  3.00
##  Median :127.00
##  Mean   : 71.77
##  3rd Qu.:127.00
##  Max.   :127.00
```

```
summary(values(holiday_rast))
```

```
##     Layer_1
##  Min.   :  1.00
##  1st Qu.:  3.00
##  Median :  4.00
##  Mean   : 60.83
##  3rd Qu.:127.00
##  Max.   :127.00
```

c. Plot each raster.. Set the scale to be `scale_fill_brewer(palette = "Spectral", direction=-1)`

HINT: Remember we have to turn them into "data.frames" for ggplot to recognize them as plot-able.

HINT HINT: Remember to check the labels of the data values to be able to set the fill.

```
rebel_rast_df <- as.data.frame(rebel_rast, xy = TRUE)
holiday_rast_df <- as.data.frame(holiday_rast, xy=TRUE)


ggplot() +
    geom_raster(data = holiday_rast_df , aes(x = x, y = y, fill = Layer_1)) +
    scale_fill_brewer(palette = "Spectral", direction=-1)
```

```
## Warning: Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using 'geom_tile()' instead.
```

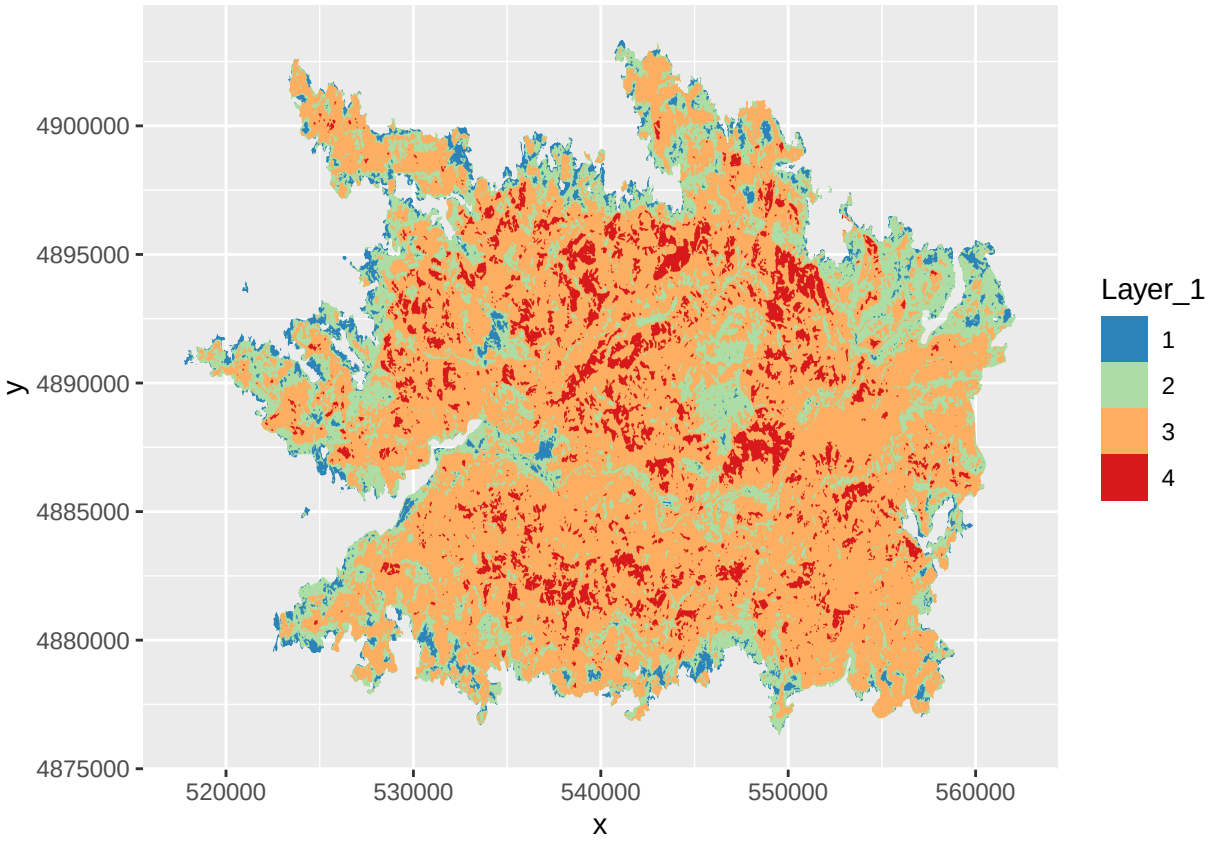


Figure 1: Holiday plot with ggplot2 using the Spectral color scale

```
ggplot() +
    geom_raster(data = holiday_rast_df , aes(x = x, y = y, fill = Layer_1)) +
    scale_fill_brewer(palette = "Spectral", direction=-1)
```

```
## Warning: Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using 'geom_tile()' instead.
```

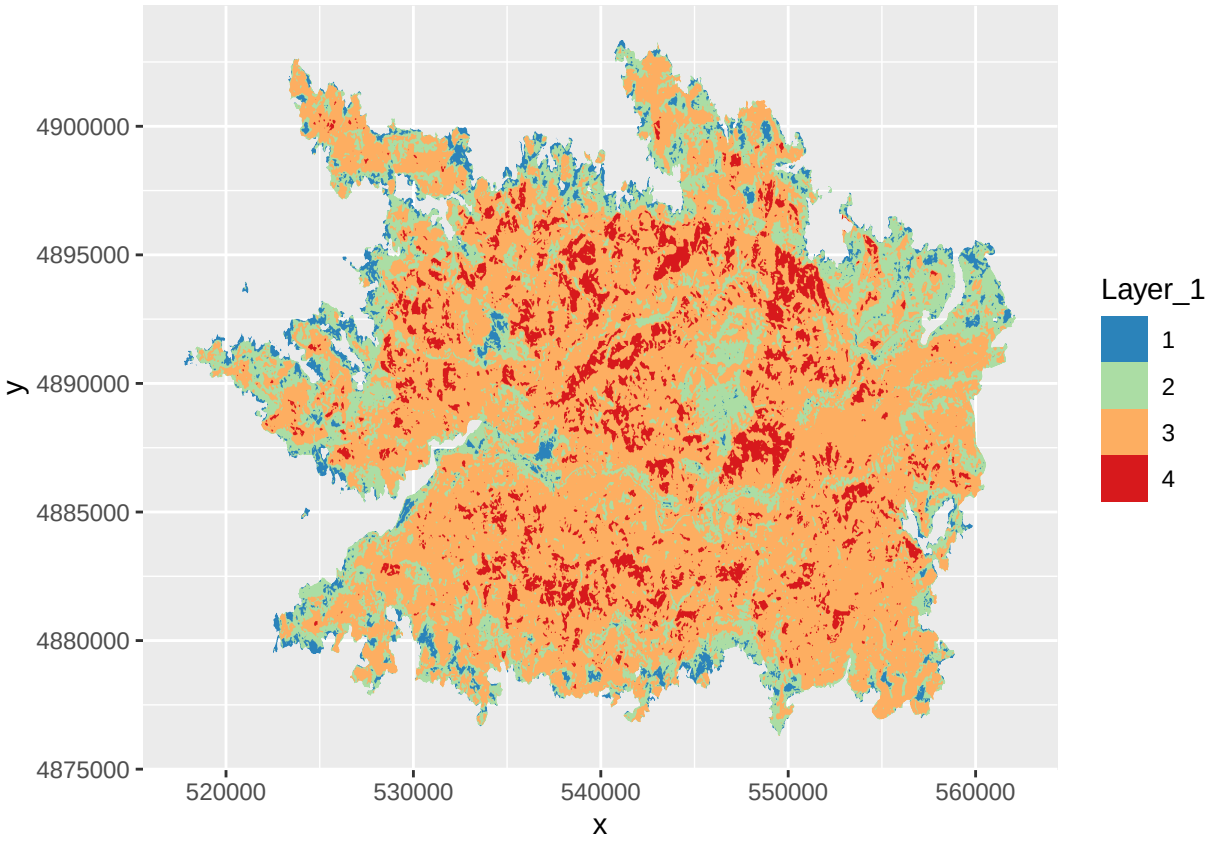


Figure 2: Holiday plot with ggplot2 using the Spectral color scale

```
beachie_rast_df <- as.data.frame(beachie_rast, xy=TRUE)

ggplot() +
    geom_raster(data = beachie_rast_df , aes(x = x, y = y, fill = Layer_1)) +
    scale_fill_brewer(palette = "Spectral", direction=-1)
```

```
## Warning: Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using ‘geom_tile()‘ instead.
```

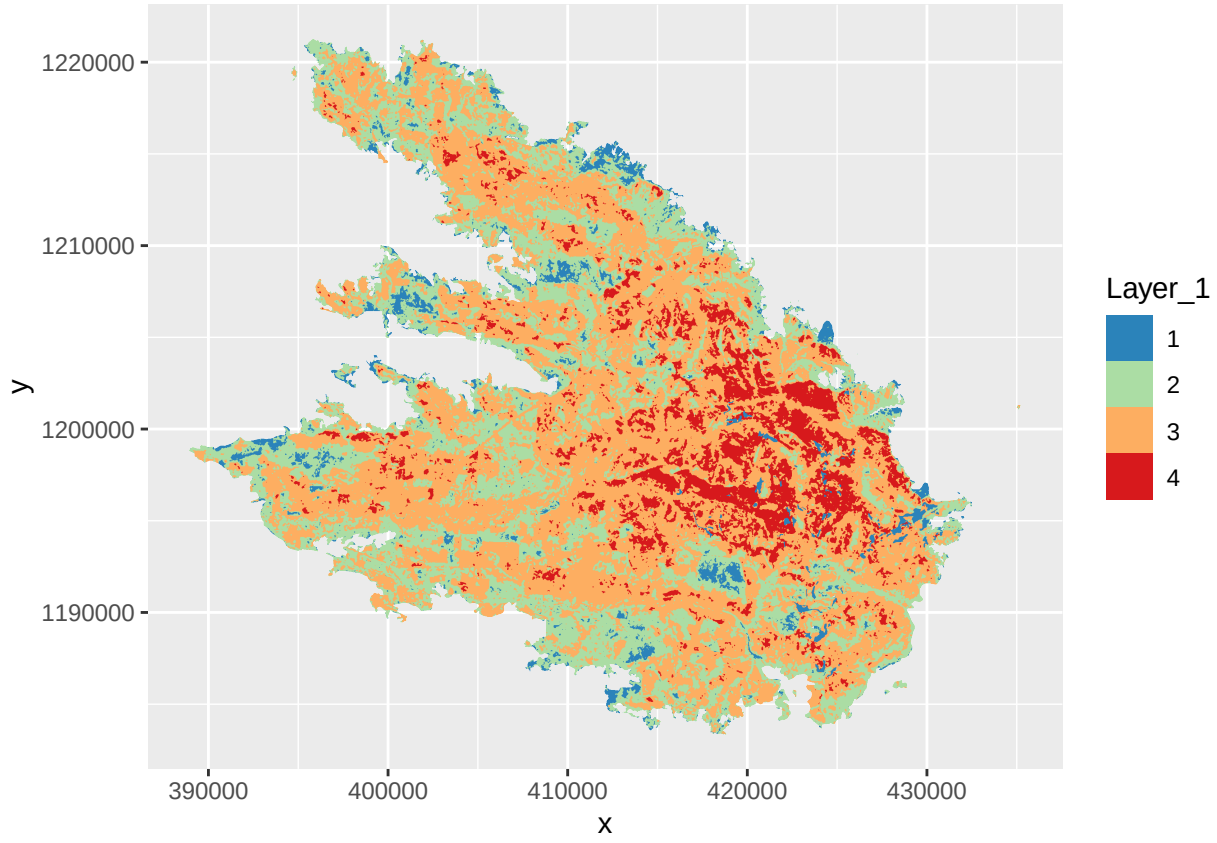


Figure 3: Beachie plot with ggplot2 using the Spectral color scale

```
terwilliger_rast_df <- as.data.frame(terwilliger_rast, xy = TRUE)

ggplot() +
    geom_raster(data = terwilliger_rast_df , aes(x = x, y = y, fill = SoilBurnSe)) +
    scale_fill_brewer(palette = "Spectral", direction=-1)
```

d. Compare these visualizations what is something you notice? -ANSWER: The areas all have varying levels of burn severity, but it definitely looks like most of the area on these visualizations are is low burn followed by unburned.

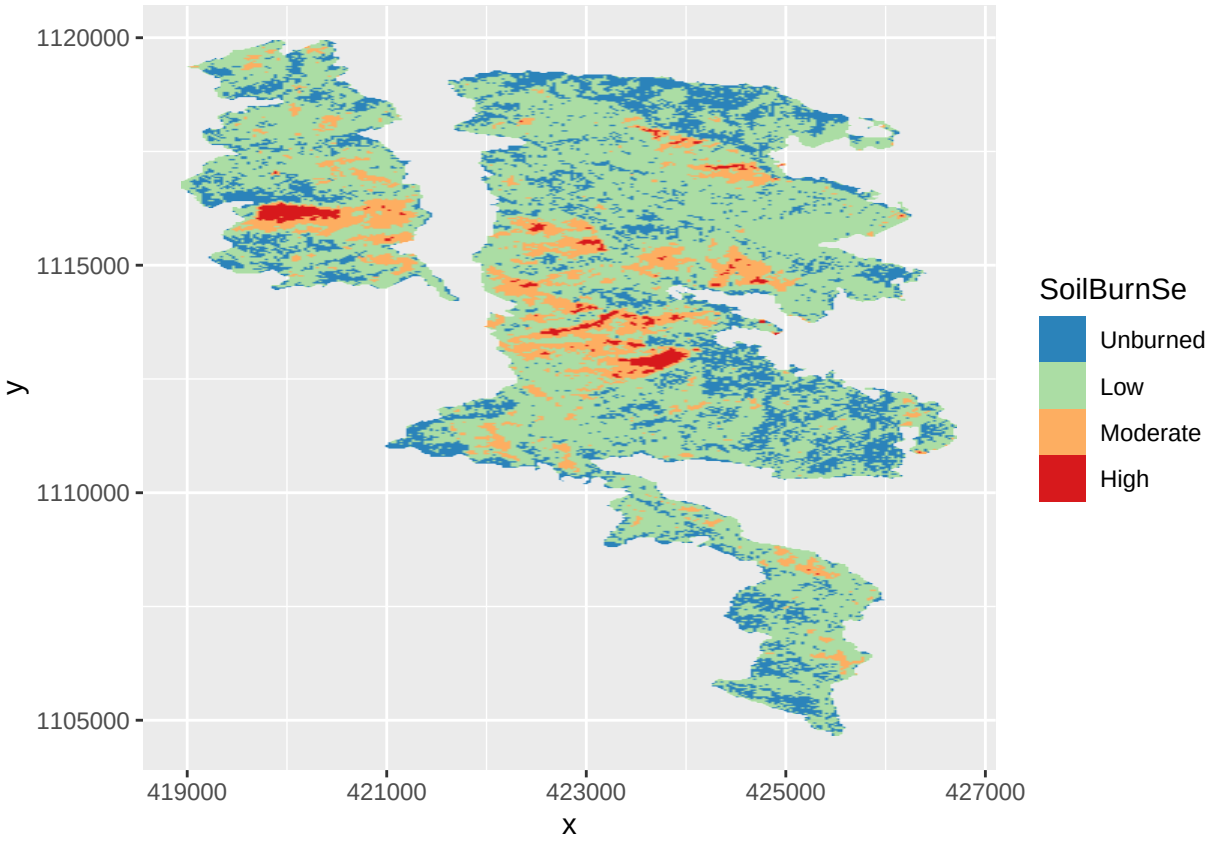


Figure 4: Terwilliger plot with ggplot2 using the Spectral color scale

Lab part 2: Exploring the attributes of our spatial data.

a. What are the crs of the rasters? What are the units? Are they all the same?

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

```
## [1] "+proj=aea +lat_0=34 +lon_0=-120 +lat_1=43 +lat_2=48 +x_0=600000 +y_0=0 +datum=NAD83 +units=m +n
```

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

```
## [1] "+proj=aea +lat_0=34 +lon_0=-120 +lat_1=43 +lat_2=48 +x_0=600000 +y_0=0 +datum=NAD83 +units=m +n
```

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

```
## [1] "+proj=utm +zone=10 +datum=NAD83 +units=m +no_defs"
```

- ANSWER crs: Holiday:"+proj=utm +zone=10 +datum=NAD83 +units=m +no_defs", datum = NAD83

Beachie:"+proj=aea +lat_0=34 +lon_0=-120 +lat_1=43 +lat_2=48 +x_0=600000 +y_0=0 +datum=NAD83 +units=m +no_defs", datum = NAD83

Terwilliger: "+proj=aea +lat_0=34 +lon_0=-120 +lat_1=43 +lat_2=48 +x_0=600000 +y_0=0 +datum=NAD83 +units=m +no_defs", datum = NAD83

- ANSWER units: Holiday:m Beachie:m Terwilliger:m
- ANSWER the same? : The same? It looks like the It looks like terwilliger and beachie creek are the same, but holiday creek has a different projection (utm instead of aea).

b. What about the resolution of each raster?

```
res(terwilliger_rast)
```

```
## [1] 30 30
```

```
res(beachie_rast)
```

```
## [1] 20 20
```

```
res(holiday_rast)
```

```
## [1] 20 20
```

- ANSWER resolution: Holiday: 30 30 Beachie: 20 20 Terwilliger: 20 20
- ANSWER the same? : The same? Holiday is different, but Beachie and Terwilliger have the same resolution.

c. Calculate the min and max values of each raster. Are they all the same?

```
minmax(terwilliger_rast)
```

```
##     SoilBurnSe
## min          1
## max          4
```

```
minmax(beachie_rast)
```

```
##     Layer_1
## min       1
## max     127
```

```
minmax(holiday_rast)
```

```
##     Layer_1
## min       1
## max     127
```

- ANSWER minmax: Holiday: Layer_1 min 1 max 127

Beachie: Layer_1 min 1 max 127

Terwilliger: SoilBurnSe min 1 max 4 - ANSWER the same? : The same? Here, Holiday and Beachie are the same but Terwilliger is different.

Given we expect there to be 4 values for each bin of severity (high, moderate, low, very low/unburned), let's try to work out why there are values other than 1-4. After checking the metadata .txt and inspecting the metadata in the raster itself, I could not find an explicit mention of the meaning on the non 1-4 data (maybe you can?). Not great practices USGS! But it is likely missing data. Let's convert the Holiday data greater than 4 to NA, just like we would a regular matrix of data.

```
holiday_rast[holiday_rast > 4] <- NA
summary(values(holiday_rast))
```

```
##     Layer_1
##  Min.   :1.0
##  1st Qu.:2.0
##  Median :3.0
##  Mean   :2.8
##  3rd Qu.:3.0
##  Max.   :4.0
##  NA's   :1536190
```

That's better :)

d. Do the same conversion for Beachie.

```
beachie_rast[beachie_rast > 4] <- NA
summary(values(beachie_rast))
```

```
##     Layer_1
##  Min.   :1.0
##  1st Qu.:2.0
##  Median :3.0
##  Mean   :2.7
##  3rd Qu.:3.0
##  Max.   :4.0
##  NA's   :2437627
```

Lab part 3: Reprojection

From our exploration above, the rasters are not in the same projection, so we will need to re-project them if we are going to be able to plot them together.

We can use the `project()` function to reproject a raster into a new CRS. The syntax is `project(RasterObject, crs)`

a. First we will reproject our `beachie_rast` raster data to match the `holidat_rast` CRS. If the resolution is different, change it to match Holiday's resolution.

Don't change the name from beachie_rast.

```
# DTM_hill_UTMZ18N_HARV <- project(DTM_hill_HARV,
#                                  crs(DTM_HARV))

#beachie_rast <- project(beachie_rast, crs(holiday_rast))


res(beachie_rast)
```

```
## [1] 20 20
```

```
res(holiday_rast)
```

```
## [1] 20 20
```

```
# getting the resolution of beachie to match Holiday's
beachie_rast <- project(beachie_rast,
                                  crs(holiday_rast),
                                  res = res(holiday_rast))

# This should return TRUE
crs(beachie_rast, proj = TRUE) == crs(holiday_rast, proj = TRUE)
```

```
## [1] TRUE
```

b. Now convert the Terwilliger crs to the holiday crs. If the resolution is different, change it to match Holiday's resolution.

```
#terwilliger_rast <- project(terwilliger_rast, crs(holiday_rast))

res(terwilliger_rast)
```

```
## [1] 30 30
```

```
res(holiday_rast)
```

```
## [1] 20 20
```

```
# getting the resolution of terwilliger to match Holiday's
terwilliger_rast <- project(terwilliger_rast,
                                  crs(holiday_rast),
                                  res = res(holiday_rast))

# This should return TRUE TRUE
crs(terwilliger_rast, proj = TRUE) == crs(holiday_rast, proj = TRUE)
```

```
## [1] TRUE
```

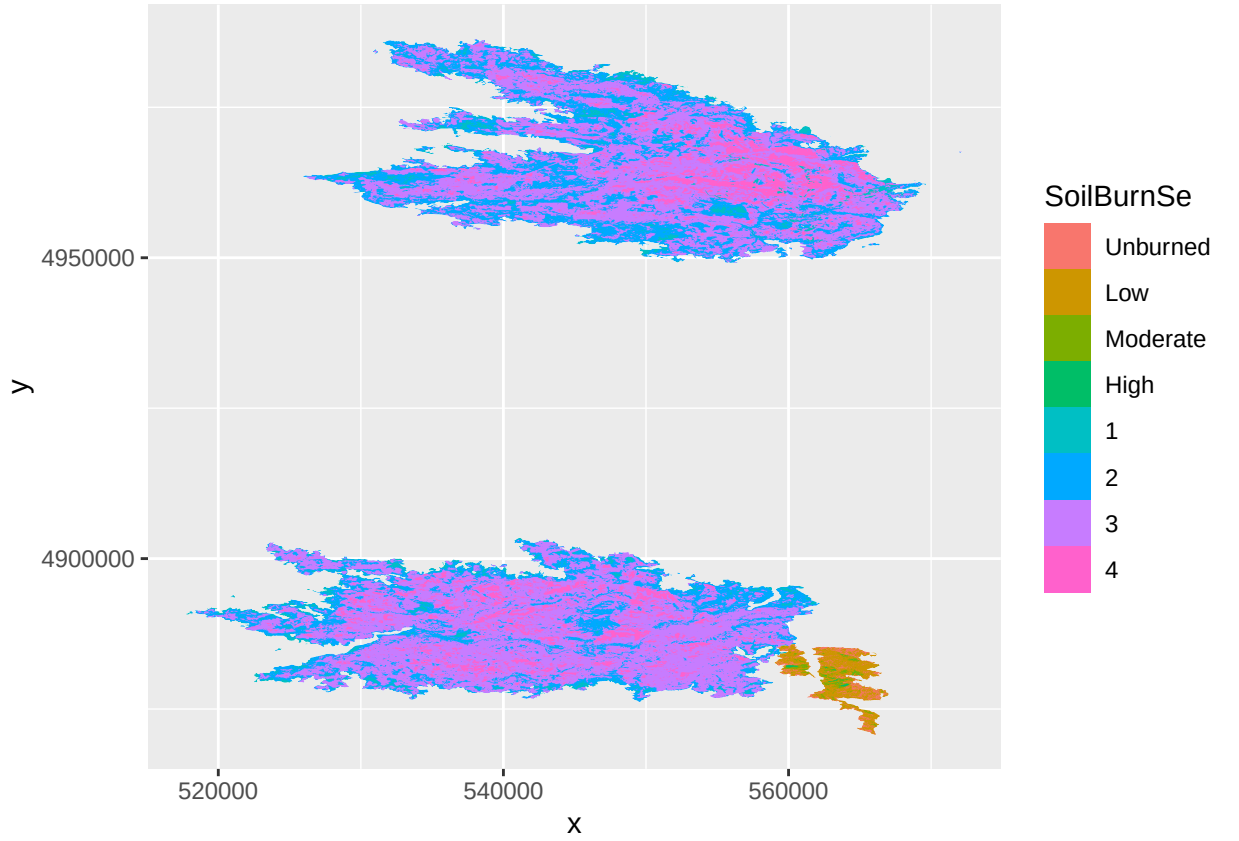
```
res(terwilliger_rast)[2] == res(holiday_rast)[2]
```

```
## [1] TRUE
```

c. Now you can plot all of the fires on the same map! HINT: Remember to re-make the dataframes.

```
# For plotting with `ggplot()`, we will need to create a dataframe from our newly
# reprojected raster. ie DTM_hill_HARV_2_df <- as.data.frame(DTM_hill_UTMZ18N_HARV, xy = TRUE)


# new dataframes!
terwilliger_rast_df_2 <- as.data.frame(terwilliger_rast, xy = TRUE)
holiday_rast_df_2 <- as.data.frame(holiday_rast, xy = TRUE)
beachie_rast_df_2 <- as.data.frame(beachie_rast, xy = TRUE)



ggplot() +
     geom_raster(data = terwilliger_rast_df_2 ,
                 aes(x = x, y = y,
                  fill = SoilBurnSe)) +
     geom_raster(data = holiday_rast_df_2,
                 aes(x = x, y = y,
                   fill = Layer_1)) +
       geom_raster(data = beachie_rast_df_2,
                 aes(x = x, y = y,
                   fill = Layer_1))
```

```
## Warning: Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using ‘geom_tile()‘ instead.
## Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using ‘geom_tile()‘ instead.
```




```
#    + scale_fill_gradientn(name = "Elevation", colors = terrain.colors(10))
```

Well that's annoying. It appears as though in 2018 the makers of these data decided to give 1,2,3,4 categorical names which are being interpreted as two different scales. If we look at the terwilliger_rast values we can see that in min max.

```
terwilliger_rast$SoilBurnSe
```

```
## class       : SpatRaster
## dimensions  : 776, 417, 1  (nrow, ncol, nlyr)
## resolution  : 20, 20  (x, y)
## extent      : 558901, 567241, 4870585, 4886105  (xmin, xmax, ymin, ymax)
## coord. ref. : NAD83 / UTM zone 10N (EPSG:26910)
## source(s)   : memory
## categories  : SoilBurnSe, BAER_Acres
## name        : SoilBurnSe
## min value   :   Unburned
## max value   :       High
```

d. Let's deal with the the easy way and modify the dataframe. Convert High to 4, Moderate to 3, Low to 2, and Unburned to 1 using your data subsetting skills.

Somethings you will need to be careful of: - If you check the class of terwilliger_rast_df$SoilBurnSe it is a factor, which is a special class of data that are ordered categories with specific levels. R will not let you

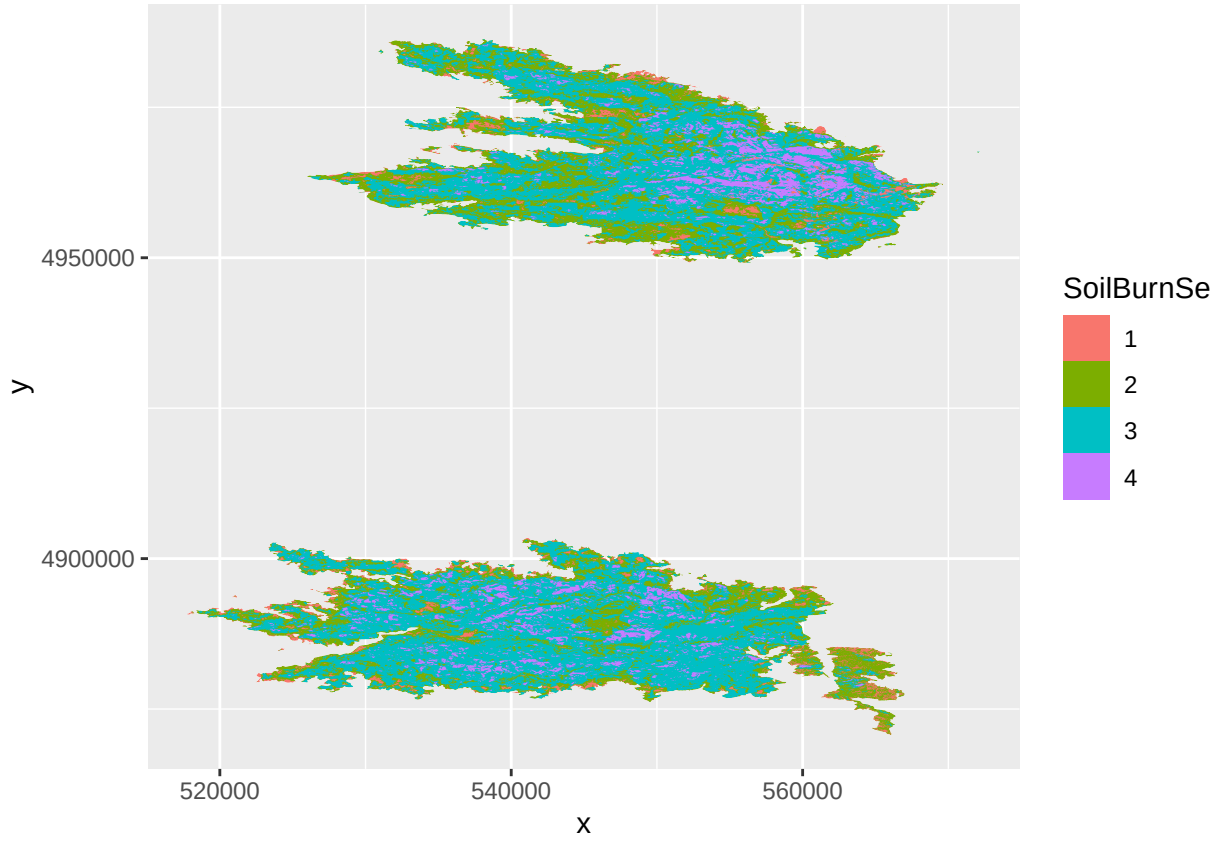
convert add a level. So first, convert the data to characters (using as.character()). - Now the data are characters, so you will not be able to add in numerics. So code the 1,2,3 as characters i.e., "1", "2"... - We will eventually want the data to be factors again so it will match up with the other rasters. So lastly, convert the data to a factor (using as.factor()).

```
# convert data to characters using as.character()
terwilliger_rast_df_2$SoilBurnSe <- as.character(terwilliger_rast_df_2$SoilBurnSe)


# bc data are now characters, code 1,2,3 as characters
terwilliger_rast_df_2$SoilBurnSe[terwilliger_rast_df_2$SoilBurnSe == "Unburned"] <- "1"
terwilliger_rast_df_2$SoilBurnSe[terwilliger_rast_df_2$SoilBurnSe == "Low"] <- "2"
terwilliger_rast_df_2$SoilBurnSe[terwilliger_rast_df_2$SoilBurnSe == "Moderate"] <- "3"
terwilliger_rast_df_2$SoilBurnSe[terwilliger_rast_df_2$SoilBurnSe == "High"] <- "4"



# convert data back to a factor using as.factor()
terwilliger_rast_df_2$SoilBurnSe <- as.factor(terwilliger_rast_df_2$SoilBurnSe)
```

e. Try plotting again.

```
ggplot() +
     geom_raster(data = terwilliger_rast_df_2 ,
                 aes(x = x, y = y,
                  fill = SoilBurnSe)) +
     geom_raster(data = holiday_rast_df_2,
                 aes(x = x, y = y,
                   fill = Layer_1)) +
       geom_raster(data = beachie_rast_df_2,
                 aes(x = x, y = y,
                   fill = Layer_1))
```

```
## Warning: Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using `geom_tile()` instead.
## Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using `geom_tile()` instead.
```




```
unique(terwilliger_rast_df_2$SoilBurnSe)
```

```
## [1] 2 1 3 4
## Levels: 1 2 3 4
```

The scale bar make sense! It would be nice to have a baselayer map to see where is Oregon these fires are.

Lab part 4: Adding in vector data

I found a nice ecoregion map on the OR spatial data website. https://spatialdata.oregonexplorer.info/geoportal/details;id=3c7862c4ae664993ad1531907b1e413e

a. Load the data into R, it is in the OR-ecoregions folder.

```
eco_region <- st_read( "OR-ecoregions/Ecoregions_OregonConservationStrategy.shp")
```

```
## Reading layer 'Ecoregions_OregonConservationStrategy' from data source
##   '/Users/miahanson/Desktop/ds-environment-MH/5-OR-fires/OR-ecoregions/Ecoregions_OregonConservationS
##   using driver 'ESRI Shapefile'
## Simple feature collection with 9 features and 6 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 183871.7 ymin: 88600.88 xmax: 2345213 ymax: 1675043
## Projected CRS: NAD83 / Oregon GIC Lambert (ft)
```

b. Check the projection and re-project if needed. We did not cover this in the lecture demo, but for vector data, use st_transform()

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

```
## [1] "+proj=lcc +lat_0=41.75 +lon_0=-120.5 +lat_1=43 +lat_2=45.5 +x_0=399999.9999984 +y_0=0 +datum=NAD
```

```
st_crs(holiday_rast, proj=TRUE)
```

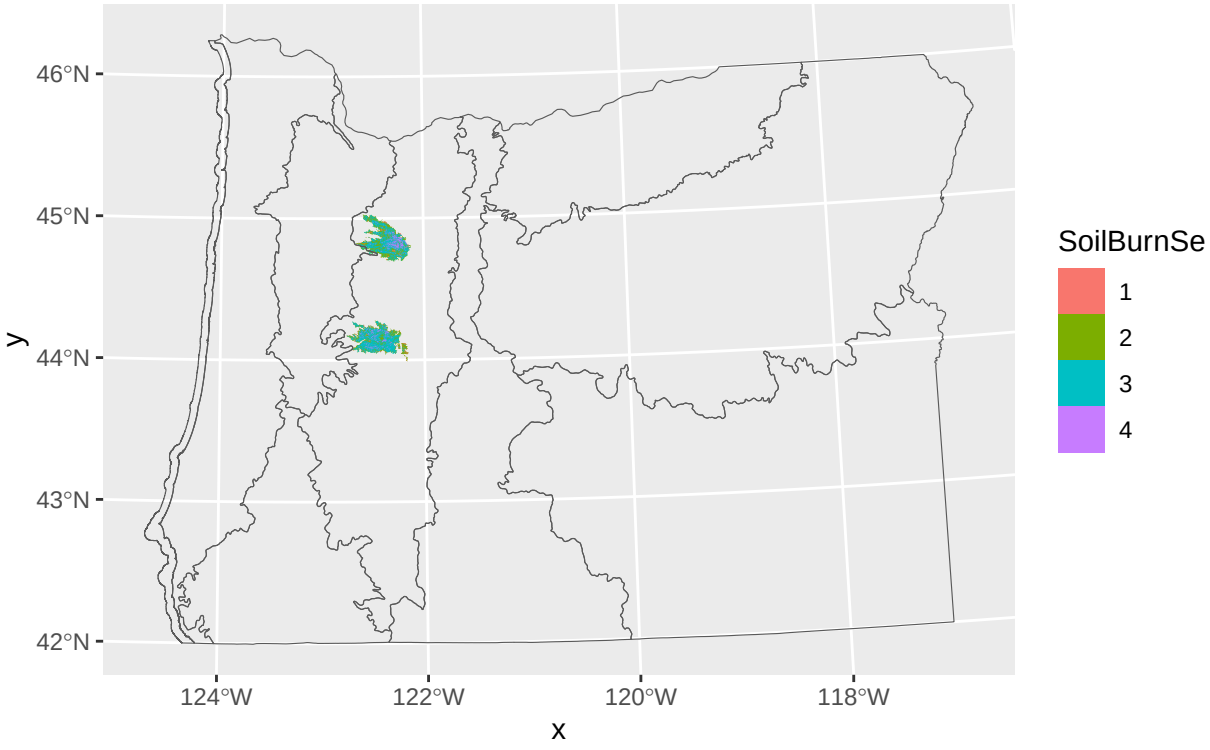
```
## Coordinate Reference System:
##   User input: NAD83 / UTM zone 10N
##   wkt:
## PROJCRS["NAD83 / UTM zone 10N",
##     BASEGEOGCRS["NAD83",
##         DATUM["North American Datum 1983",
##             ELLIPSOID["GRS 1980",6378137,298.257222101,
##                 LENGTHUNIT["metre",1]]],
##         PRIMEM["Greenwich",0,
##             ANGLEUNIT["degree",0.0174532925199433]],
##         ID["EPSG",4269]],
##     CONVERSION["UTM zone 10N",
##         METHOD["Transverse Mercator",
##             ID["EPSG",9807]],
##         PARAMETER["Latitude of natural origin",0,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8801]],
##         PARAMETER["Longitude of natural origin",-123,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8802]],
##         PARAMETER["Scale factor at natural origin",0.9996,
##             SCALEUNIT["unity",1],
##             ID["EPSG",8805]],
##         PARAMETER["False easting",500000,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8806]],
##         PARAMETER["False northing",0,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8807]]],
##     CS[Cartesian,2],
##         AXIS["(E)",east,
##             ORDER[1],
##             LENGTHUNIT["metre",1]],
##         AXIS["(N)",north,
##             ORDER[2],
##             LENGTHUNIT["metre",1]],
##     USAGE[
##         SCOPE["Engineering survey, topographic mapping."],
##         AREA["North America - between 126°W and 120°W - onshore and offshore. Canada - British Columb
##         BBOX[30.54,-126,81.8,-119.99]],
##     ID["EPSG",26910]]
```

```
?st_transform
eco_region <- st_transform(eco_region, crs= st_crs(holiday_rast))

#st_crs(eco_region)
```

c. Plot all of the data together (the rasters and vector data). You can layer on geom_sf into ggplot with the other rasters just like you would add another raster.

```
ggplot()  +
     geom_raster(data = terwilliger_rast_df_2 ,
                 aes(x = x, y = y,
                  fill = SoilBurnSe)) +
     geom_raster(data = holiday_rast_df_2,
                 aes(x = x, y = y,
                   fill = Layer_1)) +
       geom_raster(data = beachie_rast_df_2,
                 aes(x = x, y = y,
                   fill = Layer_1)) +
  geom_sf(data = eco_region, fill=NA)
```

```
## Warning: Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using ‘geom_tile()‘ instead.
## Raster pixels are placed at uneven horizontal intervals and will be shifted
## i Consider using ‘geom_tile()‘ instead.
```



We could get fancy and zoom into the correct region using extent, which we will cover next week. For now, this looks pretty good.

Lab part 5: Exploring patterns of fire severity

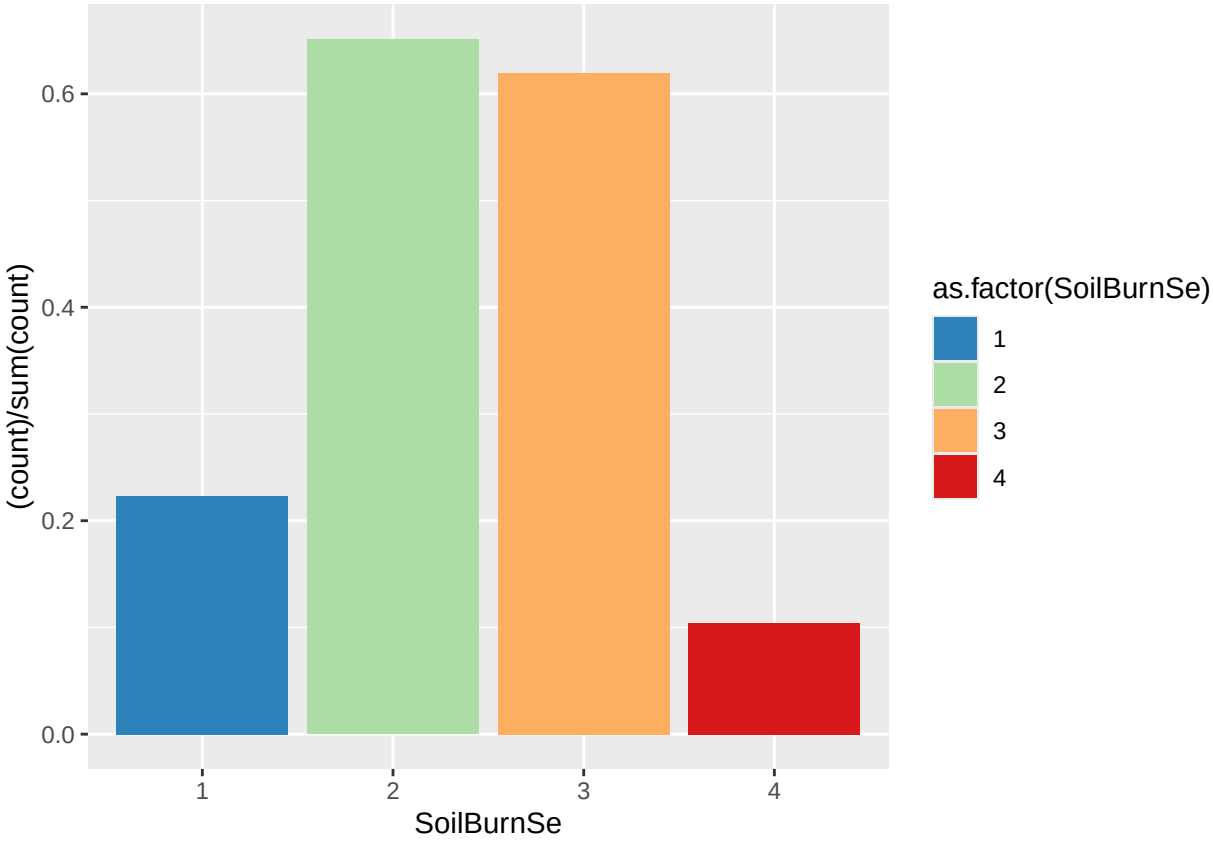
a. Create a barplot with the count of each fire severity category.

- Use scale_fill_brewer(palette = "Spectral", direction=-1) to get the bars to match the maps.
- Plot the proportion on the y. To do this, in geom_bar, include y = (..count..)/sum(..count..). EX: aes(x= Layer_1, y = (..count..)/sum(..count..)

HINT: Rather annoyingly, you will need to convert the layer values to factors again to get fill to recognize them. EX: fill=as.factor(Layer_1)

```
ggplot() +
  geom_bar(data=terwilliger_rast_df_2, aes(x=SoilBurnSe, y = (..count..)/sum(..count..), fill = as.facto
  geom_bar(data=beachie_rast_df_2, aes(x=Layer_1, y = (..count..)/sum(..count..), fill = as.factor(Layer
  geom_bar(data=holiday_rast_df_2, aes(x=Layer_1, y = (..count..)/sum(..count..), fill = as.factor(Layer
  scale_fill_brewer(palette = "Spectral", direction=-1)
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



b. What do you notice about the frequency of different severity classes when you compare these barplots. How does this relate to the Haldofsky reading? ANSWER: When I compare these barplots it looks like there's way more low and moderate occurences than the unburned and high burned areas. This relates to the Haldofsky reading because a majority of fires are still low to moderate in severity, but there has generally been an increase in higher severity fires due to climate change.

Also, if the legend label bothers you (as it does for me) Check out this tutorial: https://www.datanovia.com/en/blog/ggplot-legend-title-position-and-labels/