Spatial Data: Land Cover and Management

In the last module, we used the `sf` and `terra` packages to work with vector and raster spatial data. We calculated summary statistics by using `values()` to treat a cropped raster as a matrix. Now, we will continue doing raster calculations to explore landscape metrics across space and time.

Objectives

1. Use spatial summary statistics to calculate average agricultural quality.
2. Crop vector data and calculate land management percentages.
3. Use `freq` to calculate land cover percentages.

Getting Started

(1) Open RStudio and your R project file. (2) Create a new Quarto file.

Get your County Boundary

(3) Read the `sf` and `tigris` libraries into your Quarto document.

```
library(sf)
library(tigris)
```

(4) Get a vector of all the counties in Idaho.

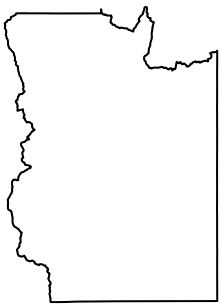
```
idaho <- counties(state = "ID", progress_bar = FALSE)
```

(5) Create a `subset` of `idaho` to create a one-row data frame for your county.

```
bear_lake <- subset(idaho, NAME == "Bear Lake")
```

(6) Plot your county using `st_geometry` to make sure it looks right.

```
plot(st_geometry(bear_lake))
```



Agricultural Land Quality

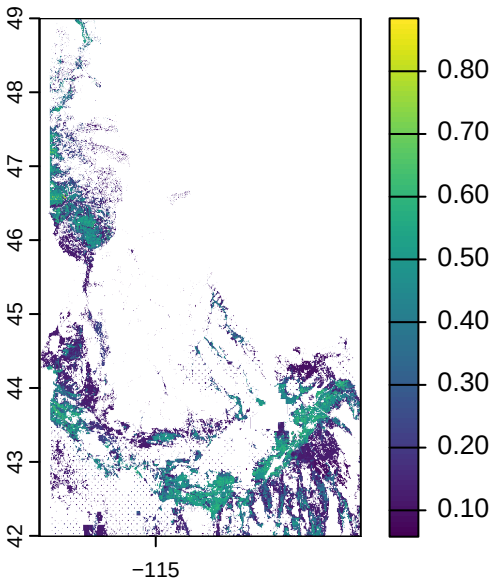
The American Farmland Trust has create a raster of agricultural land quality that takes three metrics into account: productivity, versatility, and resilience [PVR] (more information here). They combined these metrics into a unitless measure that varies from 0-1, with 1 being the highest agricultural land quality.

(7) Load the `terra` library and the PVR land quality raster (productivity_versatility_resiliency_2016_ID.tif) into your `R` session.

```
library(terra)

pvr <- rast("data/productivity_versatility_resiliency_2016_ID.tif")
```

(8) Plot the raster with `plot()`.

```
plot(pvr)
```

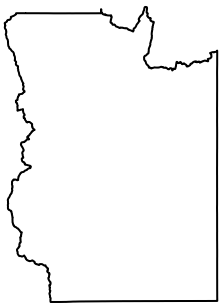


(9) Check the CRS of your county polygon. If it's not the same as the raster's CRS, use `st_transform` to re-project it.

```
my_county_proj <- st_transform(bear_lake,
                               crs = st_crs(pvr))
```

(10) `plot` the `st_geometry` of your county to make sure it looks right.

```
plot(st_geometry(my_county_proj))
```



(11) Use the `crop` function to crop the PVR raster to your county.

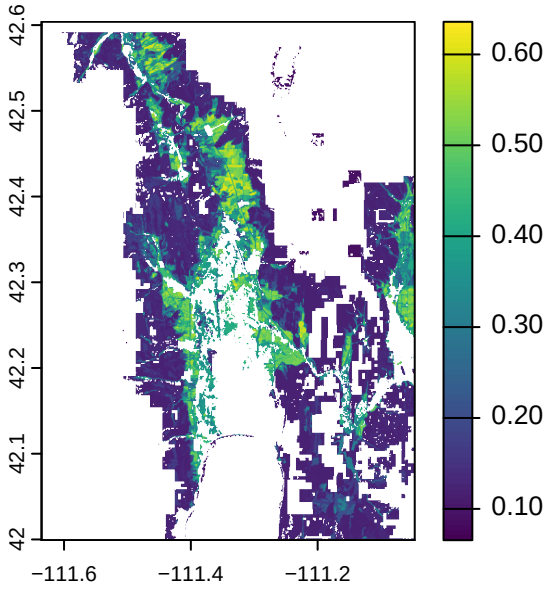
```
pvr_crop <- crop(pvr, my_county_proj)
```

(12) Mask your cropped raster to your county polygon.

```
pvr_mask <- mask(pvr_crop, my_county_proj)
```

(13) Plot your masked raster. Don't worry if it doesn't entirely fill the county. There are no agricultural quality values on developed land or federal land, and there are a lot of both (especially federal land) in Idaho.

```
plot(pvr_mask)
```



(14) Calculate the average agricultural quality for your county.

```
avg_ag_quality <- mean(values(pvr_mask), na.rm=TRUE)
```

(15) Round the average agricultural quality to two decimal places and record it in the shared Google sheet.

```
round(avg_ag_quality, digits=2)
```

```
[1] 0.22
```

Land Management

The share of land in a county that is managed by federal, state, or private entities could play a role in the type of farmland change we observe in a county. We will be using vector data from the Bureau of Land Management.

(16) Download vector data for land management from the shared Google Drive (BLM_ID_Surface_Management_Agency_Hub.zip) into your `data` folder. **(17)** Unzip the data into your `data` folder. You can unzip the data in your file explorer if that's what you're used to, or you can use the following code in your *Console*: `unzip("data/BLM_ID_Surface_Management_Agency_Hub.zip", exdir="data")`.You should

see four files with the file extensions `.shp`, `.shx`, `.dbf`, and `.prj` (there may be extra files, but these are the ones we will use in `R`).

(18) Read the data into your Quarto file with the `st_read` function from the `sf` library. You will put the file pathway to the `.shp` file, but the other files mentioned in step (17) must be in the same location as the `.shp` file, or this function will fail.

```
land_managers <- st_read("data/RLTY_SMA_PUB_24K_POLY.shp",
                         quiet=TRUE)
```

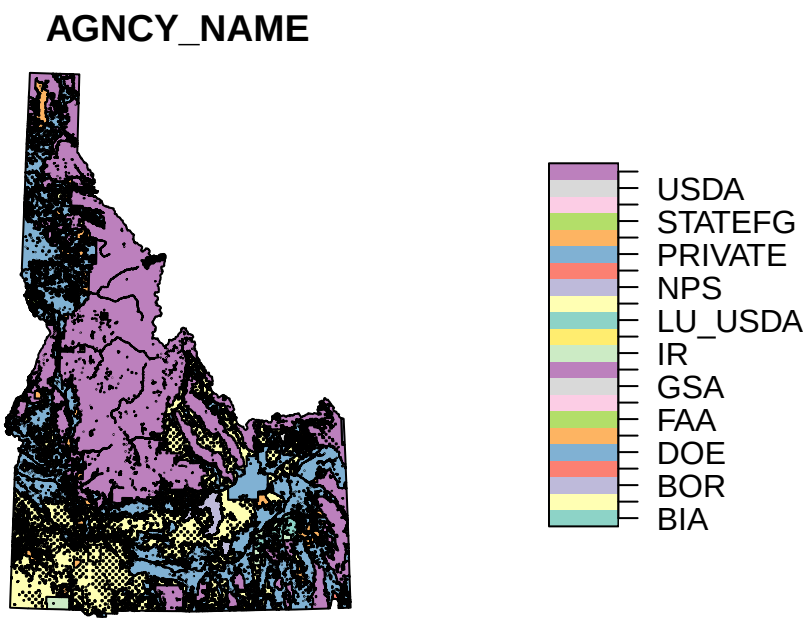
(19) Preview the data using `head`.

```
head(land_managers)
```

```
Simple feature collection with 6 features and 6 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: 2372638 ymin: 1191099 xmax: 2405296 ymax: 1196555
Projected CRS: NAD83 / Idaho Transverse Mercator
  OBJECTID MGMT_AGNCY AGNCY_NAME GIS_ACRES_ Shape_Leng Shape_Area
1        1        BLM        BLM   40.16486   1612.674   162541.4
2        2        BLM        BLM   79.16439   2405.922   320366.9
3        3        BLM        BLM   79.67083   2416.134   322416.4
4        4        BLM        BLM   78.94060   2402.913   319461.3
5        5        BLM        BLM   79.48753   2412.140   321674.6
6        6        BLM        BLM   40.42670   1617.926   163601.1
                       geometry
1 POLYGON ((2383827 1191504, ...
2 POLYGON ((2378647 1195212, ...
3 POLYGON ((2376658 1195255, ...
4 POLYGON ((2378647 1195212, ...
5 POLYGON ((2405296 1196518, ...
6 POLYGON ((2373053 1196548, ...
```

(Optional: 19.1) To get a spatial preview of the data, plot a map of the data with one of the columns as a category. If you copy this code into the Console and run it so it previews in the Plots tab, you can use the Zoom button in the Plots tab to get a full screen preview and see more detail.

```
plot(land_managers["AGNCY_NAME"])
```



To get only the land management polygons for your county, we need to find the polygons that *intersect* your county polygon using `st_intersection`. There are many more operations like this in the `sf` package. You can find a cheat sheet of examples here.

(20) Check the CRS of the `land_managers` object and your county polygon. If they do not match, use `st_transform` on your county polygon so that it matches the land management data. We transform the single polygon because it will take less time than transforming all the land management polygons. (See step (9) if needed.)

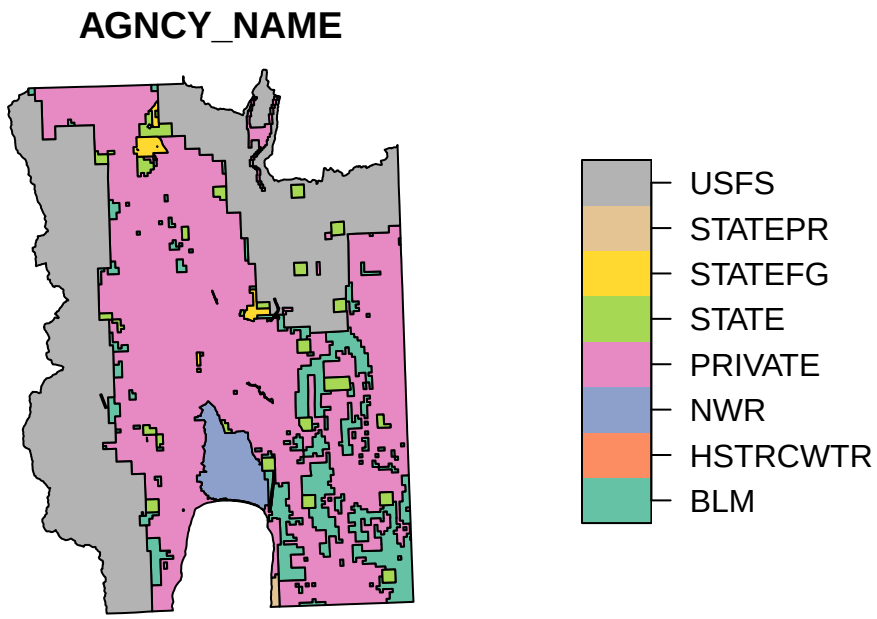
```
bear_lake_proj <- st_transform(bear_lake, st_crs(land_managers))
```

(21) Use `st_intersection` to get the land management polygons cropped to your county. The `x` argument is the object you want to extract polygons from, while the `y` object is the cropping boundary (in this case, your transformed county object).

```
bear_lake_mgmt <- st_intersection(x = land_managers,
                                  y = bear_lake_proj)
```

(22) Use `plot` to map the management agencies in your county (see step (19.1), use the AGNCY_NAME column as it has no `OTHER` category).

```
plot(bear_lake_mgmt["AGNCY_NAME"])
```



There are a lot of management agencies included in this dataset. For this project, we will put them in the following categories:

1. Private: PRIVATE
2. State: STATE, STATEFG (State Fish and Game), STATEPR (State Parks and Recreation)
3. Tribal: BIA (Bureau of Indian Affairs), IR (Indian Reservation)
4. Federal: All other codes – BLM (Bureau of Land Management), BOR (Bureau of Reclamation), COE (US Corps of Engineers), DOE (Department of Energy), DOI (Department of the Interior), FAA (Federal Aviation Administration), FHA (Federal Housing Administration), GSA (General Services Administration), LU_DOI (Bankhead Jones lands managed by the BLM), LU_USDA (National Grasslands managed by USFS), MIL (military), NPS (National Park Service), NWR (National Wildlife Service), USDA (US Department of Agriculture), USFS (US Forest Service)
5. Other: HSTRCWTR (unsurveyed water)

We used `ifelse` statements to create categories in Module 3, and we can use the same process here since we are still working with a type of `data.frame`.

(23) Create a new column in your county's land management polygons object called `Mgmt_Category` and fill it with `NA`s.

```
bear_lake_mgmt$Mgmt_Category <- NA
```

(24) If `AGNCY_NAME` is `PRIVATE`, set `Mgmt_Category` to `Private`. Else, let `Mgmt_Category` remain unchanged.

```
bear_lake_mgmt$Mgmt_Category <- ifelse(
  test = bear_lake_mgmt$AGNCY_NAME == "PRIVATE",
  yes = "Private",
  no = bear_lake_mgmt$Mgmt_Category)
```

Optional: (24.1) Use `View(your_county_mgmt)` to open your data in a new tab. Scroll to see if your `ifelse` statement correctly filled in some rows of your `Mgmt_Category` column.

```
View(bear_lake_mgmt)
```

(25) If `AGNCY_NAME` is `HSTRCWTR`, set `Mgmt_Category` to `Other`. Else, let `Mgmt_Category` remain unchanged.

```
bear_lake_mgmt$Mgmt_Category <- ifelse(
  test = bear_lake_mgmt$AGNCY_NAME == "HSTRCWTR",
  yes = "Other",
  no = bear_lake_mgmt$Mgmt_Category)
```

(26) If `AGNCY_NAME` *contains* the characters `STATE`, set `Mgmt_Category` to `State`. Else, let `Mgmt_Category` remain unchanged. (Hint: use `str_detect` from the `stringr` package in your `test`. Refer to Module 3 (34).)

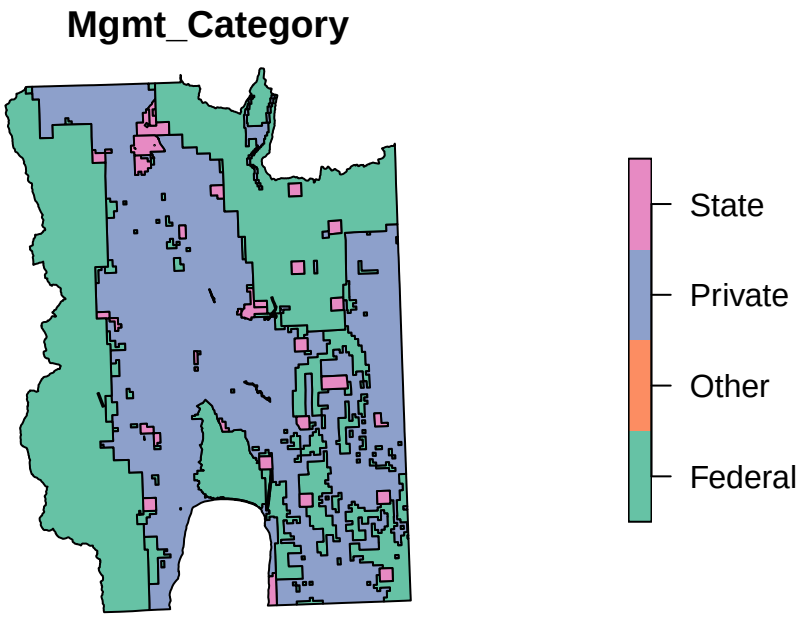
```
library(stringr)

bear_lake_mgmt$Mgmt_Category <- ifelse(
  test = str_detect(string = bear_lake_mgmt$AGNCY_NAME,
                    pattern = "STATE"),
  yes = "State",
  no = bear_lake_mgmt$Mgmt_Category)
```

(27) Finally, if `Mgmt_Category` is still `NA`, set its value to `Federal` (hint: use `is.na()` in your `test`).

```r
bear_lake_mgmt$Mgmt_Category <- ifelse(
  test = is.na(bear_lake_mgmt$Mgmt_Category),
  yes = "Federal",
  no = bear_lake_mgmt$Mgmt_Category)
```

(28) Use `plot` to map the `Mgmt_Category` column (see step (`rplot_vector_cat`‘.1). Based on your previous map, do the categories look as expected?

```r
plot(bear_lake_mgmt["Mgmt_Category"])
```



This dataset has a column called `GIS_ACRES_`, which is the acreage of each management polygon. To find the percent of county land managed by each category, we will `aggregate` the `GIS_ACRES_` column by the `Mgmt_Category` column to find the total acreage of each category, then divide by the total county acreage and multiply by 100 to get the percent of land that each category manages.

(29) Save the total acres for each management category in a new object. Print the results.

```r
mgmt_acres <- aggregate(GIS_ACRES_ ~ Mgmt_Category,
                        data = bear_lake_mgmt,
                        FUN = sum)

print(mgmt_acres)
```

```
  Mgmt_Category   GIS_ACRES_
1       Federal 1036999.6187
2         Other     144.8851
3       Private 1838387.0546
4         State   19784.6161
```

(30) Create a new column called `Percent_land` which contains the percent of your county's land that each category manages. Print your result.

```
mgmt_acres$Percent_land <- mgmt_acres$GIS_ACRES_ / sum(mgmt_acres$GIS_ACRES_) * 100

print(mgmt_acres)
```

```
  Mgmt_Category   GIS_ACRES_ Percent_land
1       Federal 1036999.6187 35.816455137
2         Other     144.8851  0.005004119
3       Private 1838387.0546 63.495208945
4         State   19784.6161  0.683331799
```

(31) Add percent management data, rounded to 1 decimal point, to the shared Google Sheet.

```
mgmt_acres$Percent_land <- round(mgmt_acres$Percent_land, digits=1)

print(mgmt_acres[, c("Mgmt_Category", "Percent_land")])
```

```
  Mgmt_Category Percent_land
1       Federal         35.8
2         Other          0.0
3       Private         63.5
4         State          0.7
```

Land Cover

(32) Download raster data for land use from the shared Google Drive

(land_cover_and_use_2016_Idaho_5070_30m.tif) into your `data` folder.

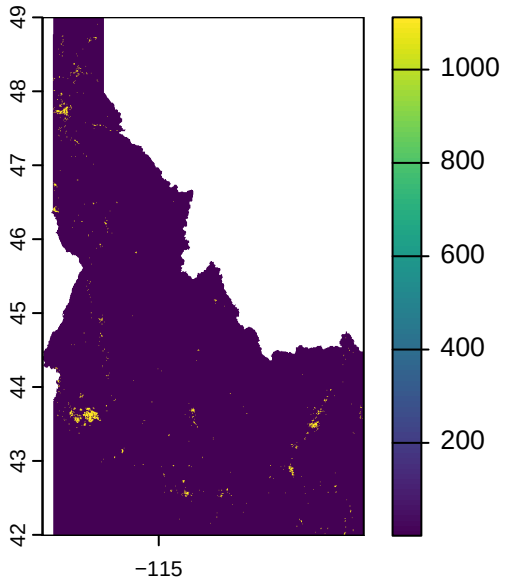
This dataset was created by American Farmland Trust (more information here). It is an agriculture specific land use map that focuses on different types of farmland and the presence of low density residential development, which is often a precursor for agricultural land loss.

(33) Read the data into your `R` session with the `rast` function.

```
land_use <- rast("data/land_cover_and_use_2016_Idaho.tif")
```

(34) Plot the raster.

```
plot(land_use)
```



In this raster, each numeric code corresponds to a land use type. The default plotting option makes this a pretty meaningless graph. However, we can't fix this until we reduce the amount of data in the raster. If we tried to manipulate this large raster with (fairly) small pixels, we might crash our `R` session. First, we'll crop this raster to our county to make analysis easier and faster.

(35) Check the CRS of the raster. If the CRS of your county polygon doesn't match the CRS of your raster, use `st_transform` to re-project it.

```
bear_lake_proj <- st_transform(bear_lake,
                               crs = st_crs(land_use))
```

(36) Use the `crop` function with the Idaho land use raster and your county polygon.

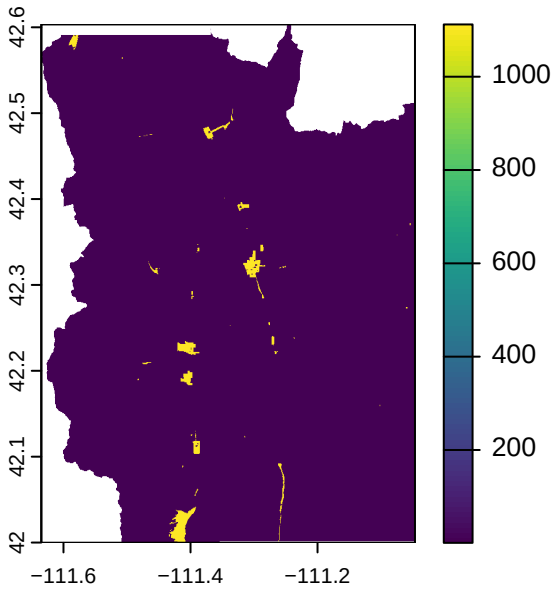
```
land_use_crop <- crop(x = land_use,
                      y = bear_lake_proj)
```

(37) Mask your *cropped* raster to your county polygon object.

```
land_use_mask <- mask(x = land_use_crop,
                      mask = bear_lake_proj)
```

(38) Plot your masked raster.

```
plot(land_use_mask)
```



Now that we've cropped our raster, it is small enough to do some calculations with. Doing calculations on large amounts of spatial data can cause `R` to crash, which is why our first step was to reduce the amount of data analyzed by cropping.

The first thing to address is the data values. This is a categorical raster where each number corresponds to a land cover. We need to manually tell `R` what those land covers are. The American Farmland Trust has a metadata file that says the land cover codes, the corresponding class, and a suggested color for each class, which is translated into the `data.frame` below.

(39) Copy the categorizing data frame below and paste it into your document.

```
land_use_cats <-
  data.frame(value = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12,
                       1101, 1102, 1103, 1104, 1105,
                       1106, 1107, 1108, 1109, 1110, 1112),
             class = c("Cropland", "Pastureland", "Rangeland",
                       "Forestland", "Woodland",
                       "Urban and highly developed areas",
                       "Water", "Federal lands without grazing",
                       "Federal lands with grazing", "Other areas",
                       "Transportation", "Cropland within LDR",
                       "Pastureland within LDR",
                       "Rangeland within LDR",
                       "Forestland within LDR",
                       "Woodland within LDR",
                       "UHD areas within LDR",
                       "Water within LDR",
                       "Federal lands without grazing within LDR",
                       "Federal lands with grazing within LDR",
                       "Other areas within LDR",
                       "Transportation within LDR"),
             color = c("#02682c", "#61aa36", "#e2c85f", "#ac9e8d",
                       "#564F46", "#601818", "#75a8ea", "#e2e2d5",
                       "#d8d7a9", "#f2f1dc", "#ab3c09", "#02682c",
                       "#61aa36", "#e2c85f", "#ac9e8d", "#564F46",
                       "#601818", "#75a8ea", "#e2e2d5", "#d8d7a9",
                       "#f2f1dc", "#ab3c09"))
```

"LDR" stands for low density residential, a type of development that can lead to more residential development and accelerated farmland loss.

The categories of a raster are called "levels" in `terra`. To apply our labels, we need to indicate that the data frame above has information about what each number means in our raster.

(40) Apply the categorization table to your cropped and masked raster by designating the `value` and `class` columns of our data frame as the levels of the raster.

```
levels(land_use_mask) <- land_use_cats[ , c("value", "class")]
```

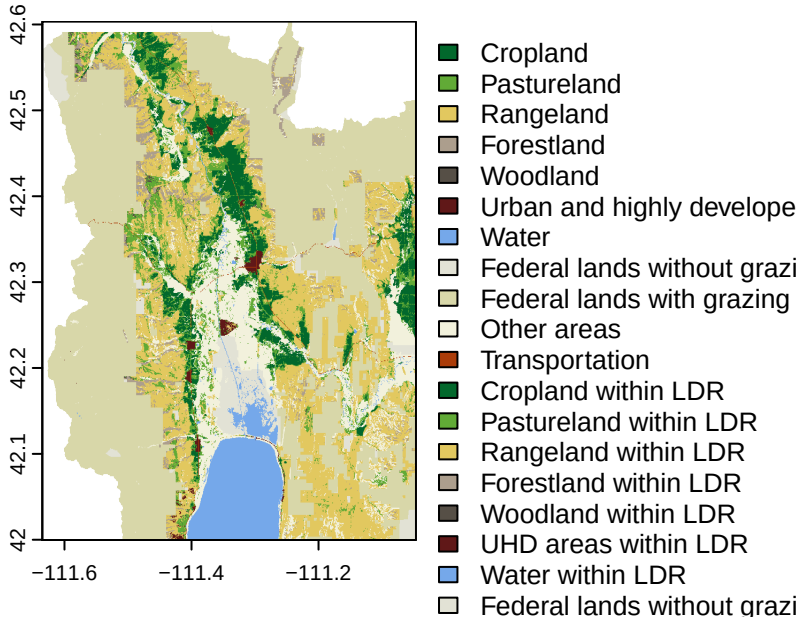
AFT also provided a suggested color range to apply to our raster to make nicer maps. Similarly to assigning a label to each number, we can assign a color to each number by designating the raster's `coltab` (color table).

(41) Designate a `coltab` for your cropped and masked raster using the `value` and `color` columns of the categorization table.

```
coltab(land_use_mask) <- land_use_cats[ , c("value", "color")]
```

(42) Plot your raster. It should be much more understandable now that we've assigned the correct metadata!

```
plot(land_use_mask)
```



Finally, we want to know the composition of land use for your county. The `freq` function calculates the frequency of each type of pixel in a raster. Dividing each category's pixel count by the total number of pixels in the raster will give us the percentage of that land cover for that county.

(43) Calculate the frequency of pixels in each category and print the result.

```
land_use_freq <- freq(land_use_mask)

print(land_use_freq)
```

```
  layer                                value   count
1     1                              Cropland  238172
2     1                           Pastureland  272973
3     1                             Rangeland  906402
```

```
4      1                                Forestland  111696
5      1                                  Woodland    8103
6      1          Urban and highly developed areas    8973
7      1                                     Water  233788
8      1             Federal lands without grazing  155375
9      1                Federal lands with grazing 1656999
10     1                               Other areas  451942
11     1                             Transportation    8454
12     1                       Cropland within LDR    4735
13     1                   Pastureland within LDR   11179
14     1                     Rangeland within LDR    6944
15     1                    Forestland within LDR     572
16     1                      Woodland within LDR     518
17     1                      UHD areas within LDR   11675
18     1                          Water within LDR      32
19     1 Federal lands without grazing within LDR      17
20     1    Federal lands with grazing within LDR       2
21     1                   Other areas within LDR    1556
22     1                Transportation within LDR    1037
```

(44) Calculate the sum of the `count` column in the frequency table you generated to find the total number of pixels and save it in an object called `total_px`.

```
total_px <- sum(land_use_freq$count)
```

(45) Create a new column in your frequency table that contains the `count` column divided by the total pixels value and multiplied by 100. This column is the percent of each land cover in your county.

```
land_use_freq$percent_cover <- land_use_freq$count / total_px *100
```

(46) Round the percent cover column to one decimal place with the `round` function and add the results to the shared Google Sheet (cropland, pastureland, rangeland, forestland, urban/developed, federal lands without grazing, federal lands with grazing, cropland in LDR, pastureland in LDR, rangeland in LDR, forestland in LDR). I've omitted some categories, but if you think they're important for your county, please let me know!

```
land_use_freq$percent_cover <- round(land_use_freq$percent_cover,
                                     digits = 1)

print(land_use_freq[, c("value", "percent_cover")])
```

```
                                    value percent_cover
1                                 Cropland           5.8
2                               Pastureland           6.7
3                                 Rangeland          22.2
4                                Forestland           2.7
5                                  Woodland           0.2
6          Urban and highly developed areas           0.2
7                                     Water           5.7
8             Federal lands without grazing           3.8
9                Federal lands with grazing          40.5
10                              Other areas          11.0
11                           Transportation           0.2
12                    Cropland within LDR           0.1
13                  Pastureland within LDR           0.3
14                    Rangeland within LDR           0.2
15                   Forestland within LDR           0.0
16                     Woodland within LDR           0.0
17                    UHD areas within LDR           0.3
18                        Water within LDR           0.0
19 Federal lands without grazing within LDR           0.0
20    Federal lands with grazing within LDR           0.0
21                  Other areas within LDR           0.0
22                Transportation within LDR           0.0
```

Finishing up

(47) At the end of your report, copy this list and paste it outside of a code chunk to create bullet points. Fill in the data you calculated for your county. Make sure that all of this information is also in the team Google Sheet for comparison with other counties. If you calculated anything extra that not on this list, be sure to add it so you have a record for later.

```
- Average agricultural quality, 2016:
- Percent federal land:
- Percent private land:
- Percent state land:
- Percent cropland, 2016:
- Percent pastureland, 2016:
- Percent rangeland, 2016:
- Percent forestland, 2016:
- Percent urban/developed, 2016:
- Percent federal lands without grazing, 2016:
```

```
- Percent federal lands with grazing, 2016:
- Percent cropland in LDR, 2016:
- Percent pastureland in LDR, 2016:
- Percent rangeland in LDR, 2016:
- Percent forestland in LDR, 2016:
```

(48) Go back through your report and add short explanations for what each code chunk does in your own words if you haven't done so already. **(49)** Render your report to a PDF and email it to [INSERT EMAIL HERE].

Statement of original and referenced work:

The entirety of this module is original work authored by Carolyn Koehn.

License