Week3 Practical

Corrections

Dark Object Subtraction

This is the process of taking the darkest reflectance value in an image and subtracting it from the all other pixels to seemingly erase atmospheric interference. This strategy has seen some push back in recent years so the process will not be covered in this practical.

Load Data and Merge

My data does not include more than one tile so the merge code will be commented out.

City Shapefile for Clip

First let's load Boise, Idaho so we can clip later.

```
library(tidyverse)
```

```
Warning: package 'purrr' was built under R version 4.5.2
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.1     v stringr   1.5.2
v ggplot2   4.0.0     v tibble    3.3.0
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.1.0
-- Conflicts ---------------------------------------- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(terra)
```

```
terra 1.8.70

Attaching package: 'terra'

The following object is masked from 'package:tidyr':

    extract
```

```
library(fs)
library(sf)
```

```
Linking to GEOS 3.13.1, GDAL 3.11.0, PROJ 9.6.0; sf_use_s2() is TRUE
```

```
library(tmap)
library(viridisLite)

ada_cities <- st_read(here::here('Week3/Week_3_Data/Cities/Cities_ADA_Surrounding.shp'))
```

```
Reading layer `Cities_ADA_Surrounding' from data source
  `C:\Users\Owen Hughes\Desktop\CASA\Remote-Sensing\GitProjects\Remote-Sensing\Week3\Week_3_D
  using driver `ESRI Shapefile'
Simple feature collection with 13 features and 9 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -116.7233 ymin: 43.08667 xmax: -115.6577 ymax: 43.92994
Geodetic CRS:  NAD83
```

Now inspect the cities.

```
qtm(ada_cities)
```

In the large cluster of cities towards the north east, the largest city is Boise so that's the city I want to select now.

```
boise <- ada_cities %>%
  filter(NAME == "Boise City")

qtm(boise)
```

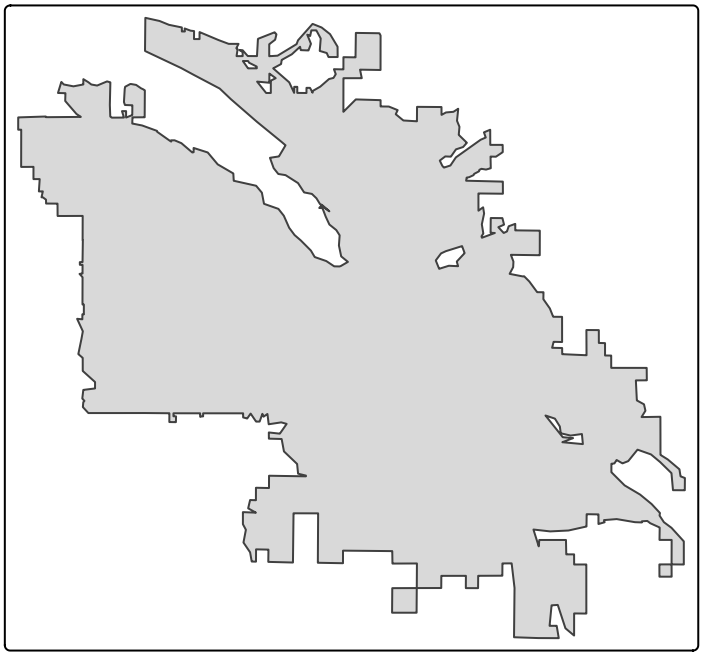


Image Stack

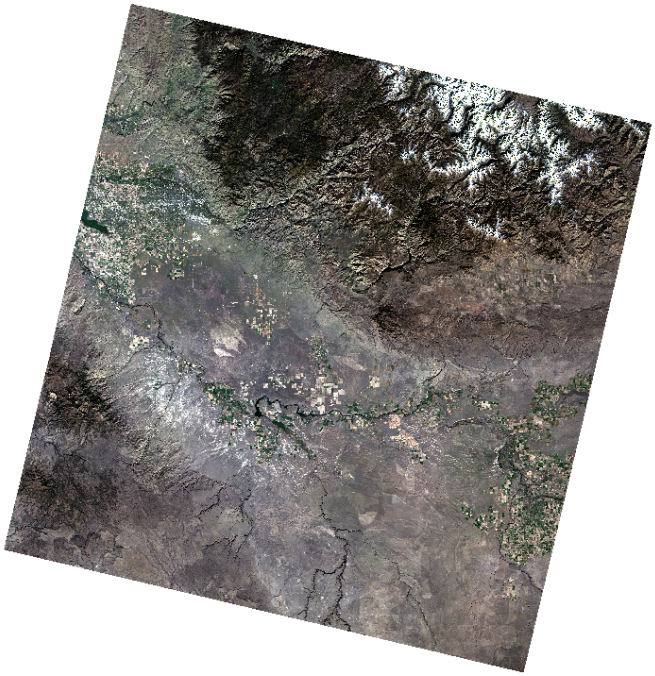
The next step is to load the multiple bands of the satellite image into a single stacked image.

```
path <- here::here("Week3/Week_3_Data/images")

tifs <- list.files(path, pattern = "B[1-7]\\.TIF$", full.names = TRUE)

landsat <- rast(tifs)
landsat
```

```
class       : SpatRaster
size        : 7821, 7701, 7  (nrow, ncol, nlyr)
resolution  : 30, 30  (x, y)
extent      : 494085, 725115, 4664385, 4899015  (xmin, xmax, ymin, ymax)
coord. ref. : WGS 84 / UTM zone 11N (EPSG:32611)
sources     : LC08_L2SP_041030_20251021_20251120_02_T1_SR_B1.TIF
              LC08_L2SP_041030_20251021_20251120_02_T1_SR_B2.TIF
              LC08_L2SP_041030_20251021_20251120_02_T1_SR_B3.TIF
              ... and 4 more sources
names       : LC08_~SR_B1, LC08_~SR_B2, LC08_~SR_B3, LC08_~SR_B4, LC08_~SR_B5, LC08_~SR_B6,
```

Check a true color composite image

```
plotRGB(landsat, r = 4, g = 3, b = 2, stretch = 'lin')
```



Clip image

Now I want to only focus on a small area because the size of the image may delay processing. This small area will be the outline of Boise saved earlier.

Check the crs of both areas.

```
print(crs(landsat))
```

```
[1] "PROJCRS[\"WGS 84 / UTM zone 11N\",\n    BASEGEOGCRS[\"WGS 84\",\n        DATUM[\"World
```

```
print(crs(boise))
```

```
[1] "GEOGCRS[\"NAD83\",\n    DATUM[\"North American Datum 1983\",\n        ELLIPSOID[\"GRS 19
```

Change the Boise shapefile to the same crs as the satellite image.

```
boise <- st_transform(boise, "EPSG:32611")
crs(boise)
```

```
[1] "PROJCRS[\"WGS 84 / UTM zone 11N\",\n    BASEGEOGCRS[\"WGS 84\",\n        ENSEMBLE[\"Wor
```

Clip the image to Boise.

```
landsat_clip<-landsat%>%
  terra::crop(., boise)%>%
  terra::mask(., boise)
```

Check to see if the clip was successful.

```
plotRGB(landsat_clip, r = 4, g = 3, b = 2, stretch = 'lin')
```

Sweet Looks Good

Ratio

We can use ratios of spectral signatures to take advantage of the fact that different materials have different reflectance in different bands. For instance, healthy vegetation has a **much** higher reflectance in the near infrared band than the red band. This stark change in reflectance can be leveraged to display areas of healthy vegetation more clearly on a satellite image.

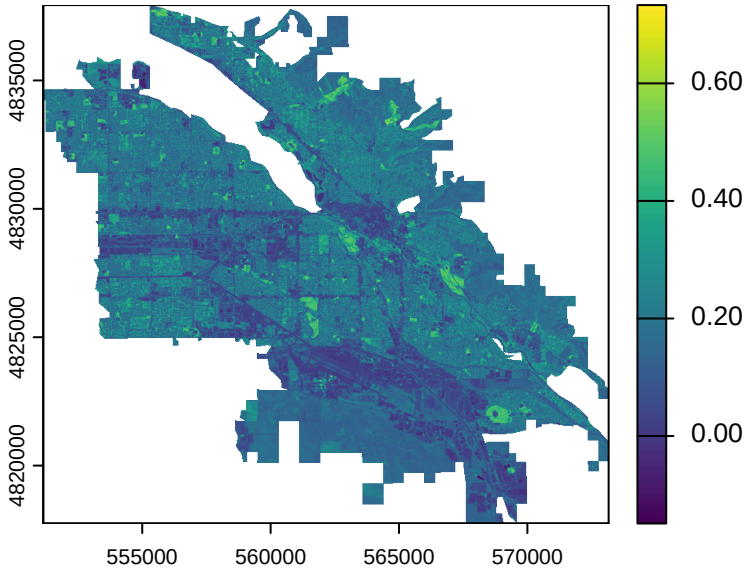
NDVI

This ratio is called the Normalize Difference Vegetation Index and is calculated with this formula:

$NDVI = [NIR - Red]/[NIR + Red]$

Let's put this into practice with our Boise example!

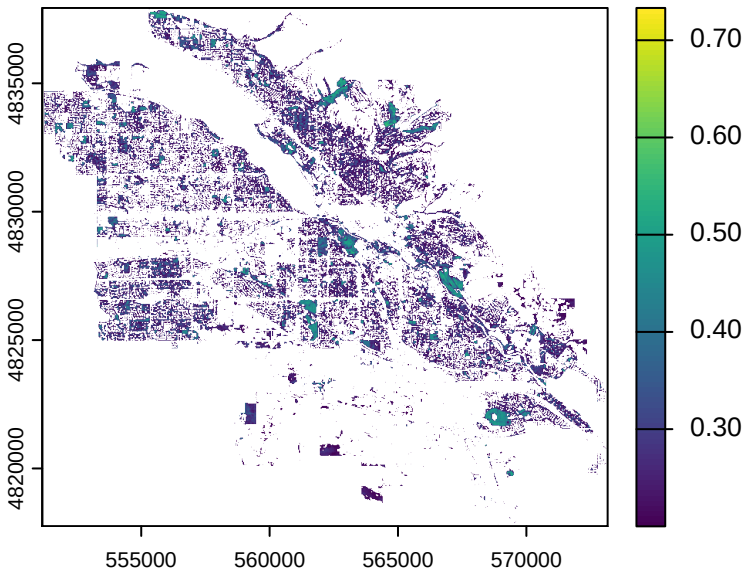
```
boise_NDVI <- (landsat_clip$LC08_L2SP_041030_20251021_20251120_02_T1_SR_B5 - landsat_clip$LC

boise_NDVI %>%
  plot(.)
```



Perfect! As one might expect, the city itself is not full of healthy vegetation and the fact that this image if from the winter doesn't help either. However, we can see several large areas within the city with noticeable higher rates of healthy vegetation. These are Boise's famous city parks... well... and a several courses. In the southwest we can see an ovular shaped park, this is the Simplot Sports Complex, in Downtown Boise (near the white enclave (Federal owned land not part of the city) in the NW quadrant) we can see the famous triply connected parks of Katherine Albertson, Anne Morrison, and Julia Davis parks, and int he the north many affluent hillside communities have their own semi-private parks and golf courses.

But! What happens if we filter for areas only with an NDVI ratio higher than 0.2? This should really show the parks and golf courses.

```r
veg <- boise_NDVI %>%
  # cbind = combine dataframes, or here or values listed
  terra::classify(., cbind(-Inf, 0.2, NA))

veg %>%
  plot(.)
```



Yep, we can definitely see the parks quite clearly now, however at this level we can also see another interesting cultural fact about Boise. Many desert and savanah cities in the U.S (of which Boise is included), put an emphasis on irrigated lawns and gardens. This means that on this map, the only areas that are completely left out are the airport, industrial areas, and the grassy hills (which in the winter have no live vegetation).

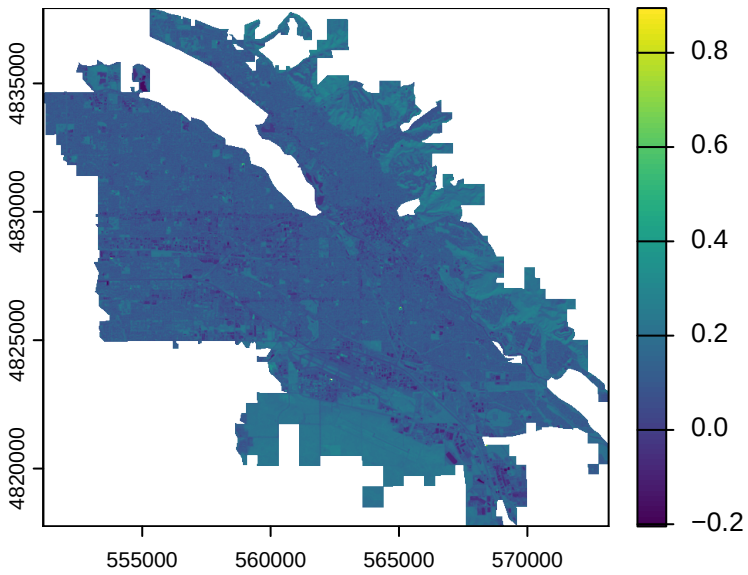
NDSI (soil)

Now lets try out a another random ratio to see the result. Boise has been on of the fastest growing metropolitan areas in the United States since 2000 (by percent population change) and as a result, the City of Boise has annexed much land development. These annexations tend to occur in blocky chunks, hence why the outline of the city (particularly in the north an south) looks pixelated. So… why don't we see if any of this annexed land (and the whole city more broadly) is still yet to be developed as of January 2026.

The Normalized Difference Soil Index will show areas that are neither built up or irrigated, and is defined by this equation

$NDSI = [SWIR1 - Green]/[SWIR1 + Green]$

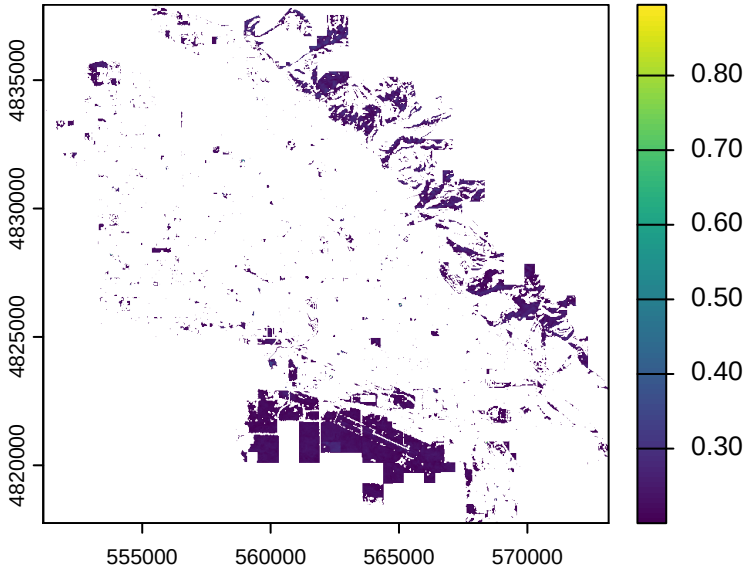
```
boise_NDSI <- (landsat_clip$LC08_L2SP_041030_20251021_20251120_02_T1_SR_B6 - landsat_clip$LC

boise_NDSI %>%
  plot(.)
```



Just as I suspected! It looks like the areas with the highest bare soil indexes are in the northwest and south which have been annexed most recently. Another note is that the southern area also contained the airport, so much of that bare soil will be from areas next to the runways.

Let's limit the map to display areas with bare soil ratios only over 0.2.

```
soil <- boise_NDSI %>%
  # cbind = combine dataframes, or here or values listed
  terra::classify(., cbind(-Inf, 0.2, NA))

soil %>%
  plot(.)
```



And Bob's your uncle! The airport is clearly visible in the south, and the hilly areas yer to be developed (or impossible to develop on) in the north east are clearly the main areas of bare soil.

Filtering

Filtering is a way to change the value of a raster pixel to smooth images, detect variance, detect intensity changes, etc. Let's try applying a simple 3x3 filter to the red band. This means that each cell uses a 3x3 square of pixels around it to approximate a new value.

Sum v. Mean

This next code block uses sums of the surrounding cells meaning that the new value will be roughly 9x higher than the original.

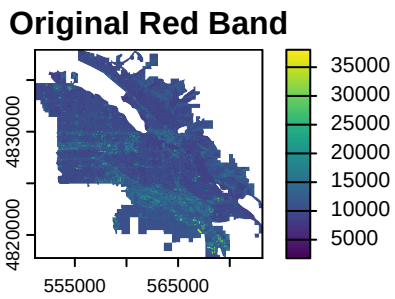
```
boise_filter1 <- terra::focal(landsat_clip$LC08_L2SP_041030_20251021_20251120_02_T1_SR_B4, w=
```

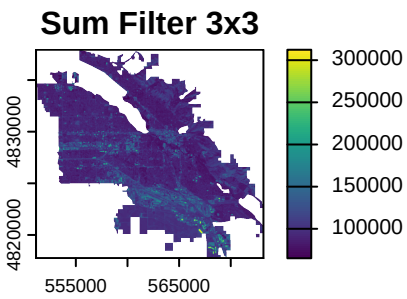
This cell block forces the use of mean filtering meaning that the min and max of the filtered raster will be similar to the original.

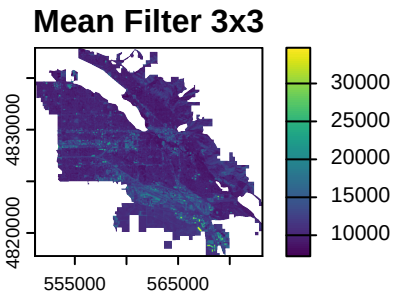
```r
boise_filter2 <- focal(
  landsat_clip$LC08_L2SP_041030_20251021_20251120_02_T1_SR_B4,
  w = matrix(1/9, 3, 3),    # explicit mean kernel
  fun = sum                 # sum of weights = 1, so this is a mean
)
```

Let's give these a plot shall we.

```r
par(mfrow = c(2, 2))

plot(
  landsat_clip$LC08_L2SP_041030_20251021_20251120_02_T1_SR_B4,
  main = "Original Red Band",
)

plot(
  boise_filter1,
  main = "Sum Filter 3x3",
)

plot(
  boise_filter2,
  main = "Mean Filter 3x3",
)

par(mfrow = c(1, 1))
```



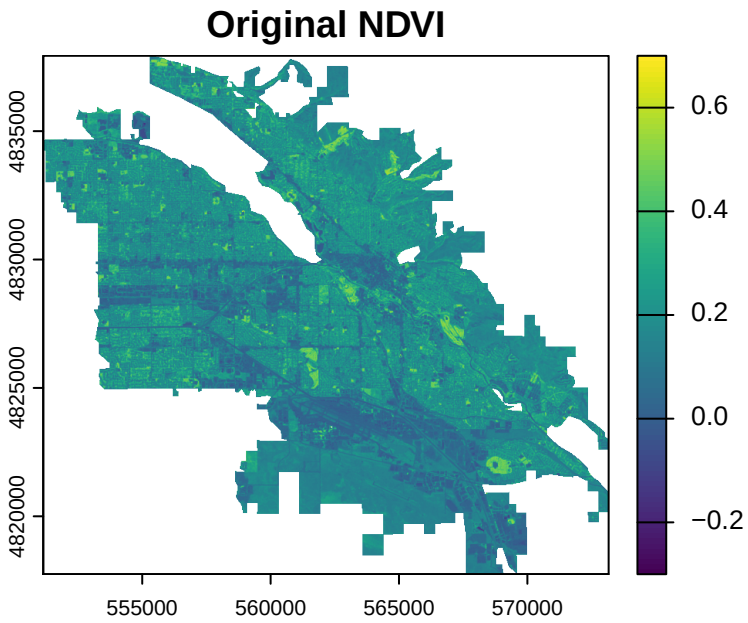




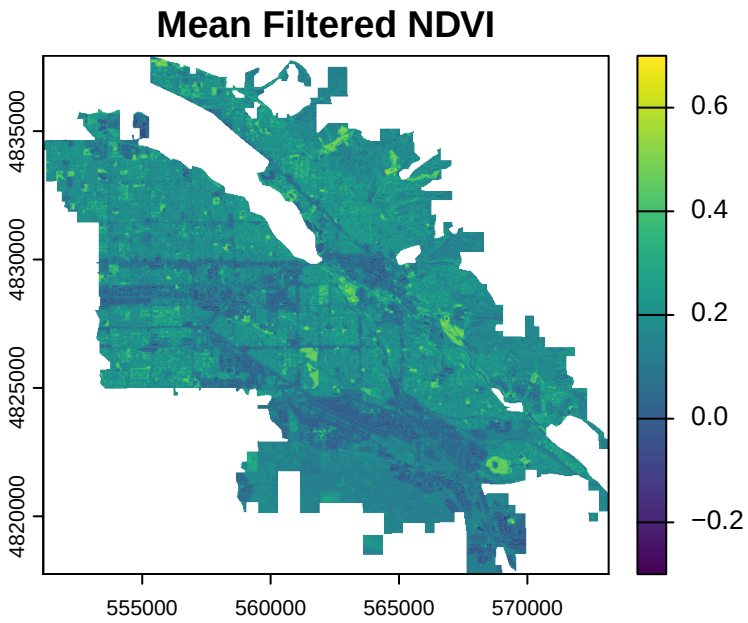
NDVI Plots

Let's see what happens if we try the NDVI again using the filter.

```
boise_NDVI1 <- (landsat_clip$LC08_L2SP_041030_20251021_20251120_02_T1_SR_B5 - boise_filter2)

boise_NDVI2 <- (landsat_clip$LC08_L2SP_041030_20251021_20251120_02_T1_SR_B5 - boise_filter1)

plot(
  boise_NDVI,
  main = "Original NDVI",
  col = viridis(100),
  range = c(-0.3, 0.7)
)
```

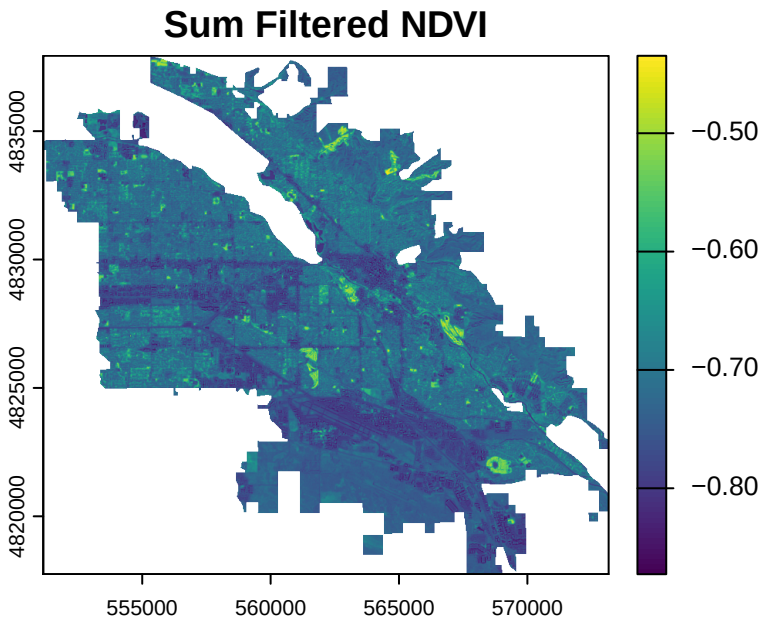



```
plot(
  boise_NDVI1,
  main = "Mean Filtered NDVI",
  col = viridis(100),
  range = c(-0.3, 0.7)
)
```



These plots are effectively the same, meaning that the filter we completed did not make any substantial changes. But let's check the sum filter, it's scale will be different as the cell values were generally higher.

```
plot(
  boise_NDVI2,
  main = "Sum Filtered NDVI",
  col = viridis(100),
  #range = c(-0.3, 0.7)
)
```



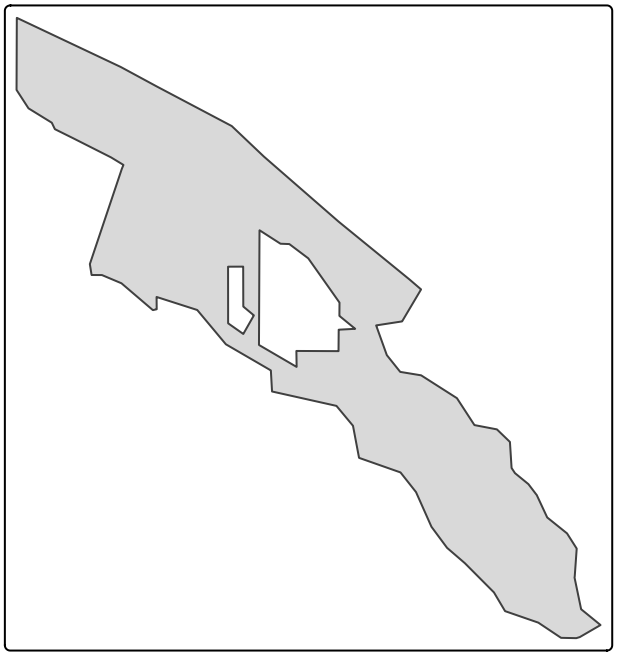
Yep. We see that the scale bar is different and the raster cells have different values in the sum filtered raster. However, the larger result is still the same. I can still easily pick out the parks, golf courses, and areas with little vegetation. This is because the filters we applied are low-pass smoothing filters that maintain general trends rather than display variance or another more extreme change,

Texture

Texture is a way for us to almost categorize certain materials and display them in a way that emphasizes their importance for our particular use case. For this example, we will be changing cities as Boise is quite large in area and these texture functions can take a while to run. So instead of Boise, we will focus on Garden City. You may notice that the north west of Boise has an elongated protrusion that pokes into the city and splits the north of the city into a west and eastern area. This protrusion is Garden City.

Extract new city

```
gc <- ada_cities %>%
  filter(NAME == "Garden City")

qtm(gc)
```



```
gc <- st_transform(gc, "EPSG:32611")
crs(gc)
```

```
[1] "PROJCRS[\"WGS 84 / UTM zone 11N\",\n    BASEGEOGCRS[\"WGS 84\",\n        ENSEMBLE[\"Wor
```

```
gc_clip<-landsat%>%
  terra::crop(., gc)%>%
  terra::mask(., gc)

plotRGB(gc_clip, r = 4, g = 3, b = 2, stretch = 'lin')
```

Within Garden city we can see an enclave. This ironically is the Ada County owned Fair Grounds. Despite this, Garden city still has a golf course, part of the Boise River, neighborhood, and parks so we should still be able to attain useful results from the texturing.

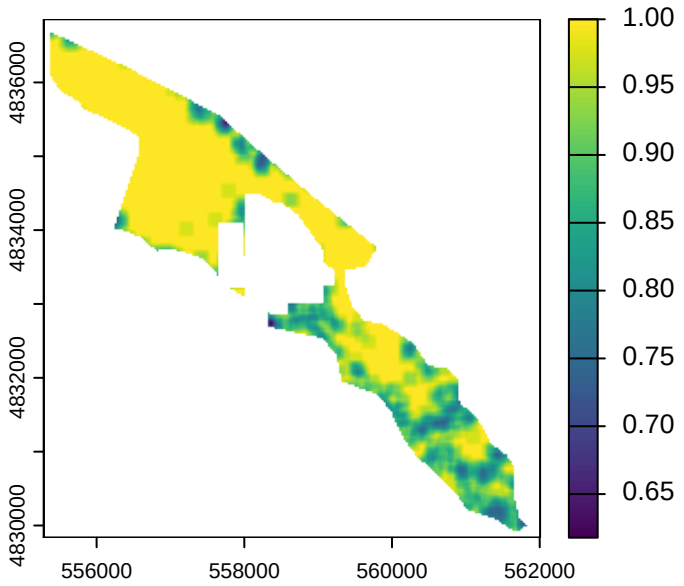
Texturing Code

```
library(GLCMTextures)
```

```
Warning: package 'GLCMTextures' was built under R version 4.5.2
```

```
scale <-(gc_clip*0.0000275) + -0.2

textures1<- glcm_textures(
  scale$LC08_L2SP_041030_20251021_20251120_02_T1_SR_B4,
  # size of window
  w = c(7,7),
  # levels means divide the data into 4 "bins" e.g. a range of 0-20
  # would be 0-5, 5-10, 10-15,15-20
  n_levels = 4,
  # raster data might not be greater than 0
  # convert it to a discrete number of grey levels (e.g. 4)
```

```
  # the data is equally divided between the 4 levels by using "range"
  quant_method = "range",
  #co-occurence (second order) matrix (1,0) = one pixel to the right
  # default is all directions as below
  shift = list(c(1, 0), c(1, 1), c(0, 1), c(-1, 1)),
  # select what we want
  metrics="glcm_homogeneity")

plot(textures1)
```



This very simple example understandably doesn't tell us much, however generally, the areas with lower values are the more industrial and bare soil areas of Garden City, while the higher values are in the more green/neighborhod areas of Garden city.

Data Fusion

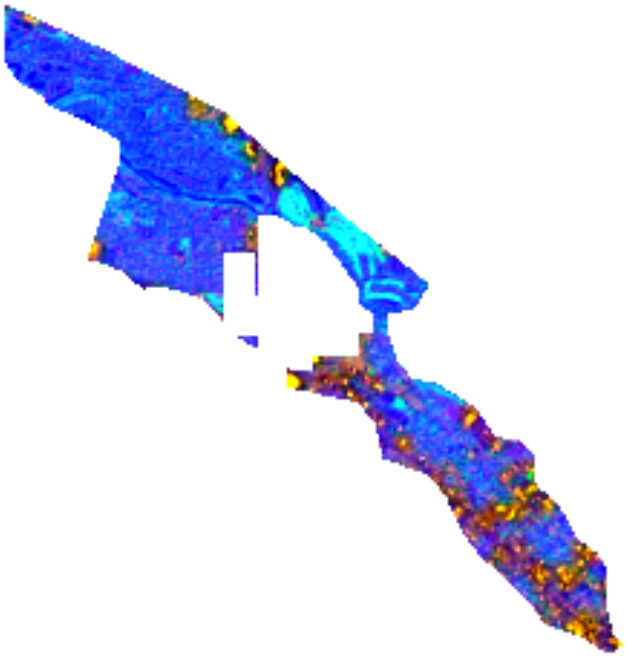
Data fusion allows us to mix R.S. and art by combing our research methods with creating a product to view. This becomes more subjective and the solution to a problem is often down to the analyst.

For our example, I am going to clip the recently made texture data back to the original satellite imagery.

```
raster_and_texture <- c(gc_clip, textures1)
```

False Color Composit

```
plotRGB(raster_and_texture, r=4, g=5,
        b=nlyr(raster_and_texture), stretch="lin")
```



Using this false color composite, we can see where the texture analysis worked its magic. With the red color represented by the red band, the green color represented by NIR, and the blue color represented by the texture band, the grassy golf course becomes evident in cyan; I mean we can even see individual holes. The bare soil and large building areas are very visible in red/yellow/orange and the neighborhoods, river, and roads are all blue.

PCA

PCA is a way to reduce dimensionalality in our data that facilitates comparisons between data that isn't measured in the same way. For example, out textural data and our spectral data! There are two main components: centering and scaling.

- Centering subtracts the mean of the variable from each data point

- Scaling divides the data by the standard deviation

Let's give it a go!

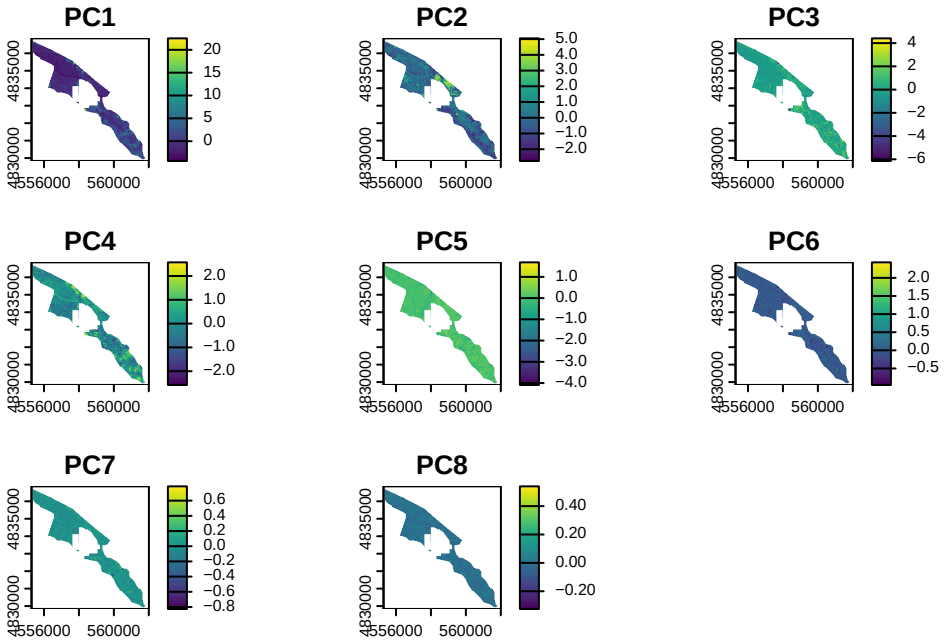
```
pca <- prcomp(as.data.frame(raster_and_texture, na.rm=TRUE),
              center=TRUE,
              scale=TRUE)

summary(pca)
```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5    PC6     PC7
Standard deviation     2.4459 1.0363 0.68657 0.61829 0.23092 0.1650 0.08893
Proportion of Variance 0.7478 0.1342 0.05892 0.04778 0.00667 0.0034 0.00099
Cumulative Proportion  0.7478 0.8820 0.94096 0.98874 0.99541 0.9988 0.99980
                           PC8
Standard deviation     0.04013
Proportion of Variance 0.00020
Cumulative Proportion  1.00000
```

PCA takes multicolinearity into account ensuring that independent variables are not correlated (sometimes creating new variables to ensure this). Though, it can be difficult to make meaningful conclusions about the data especially if there are several variables in the component analysis.

Map the PCA

```
x <- predict(raster_and_texture, pca)
plot(x)
```



Because satellite bands can be highly correlated, PCA serves as a way to differentiate between them in analysis by reducing redundant information, displaying certain features more prominently, and creating more visual contrast.