wk05-01_inclass

Patrick Bitterman

2025-02-09

Week 05.01 in-class

Intro

Today's exercise is a bit different. This notebook will demonstrate some new techniques while also allowing you to become more comfortable with the R Markdown (Rmd) format. You can click the little green arrow for each of the code blocks to run everything IN THAT BLOCK.

First, let's add the packages we'll need. NOTE, you may also need to install some of these packages onto your computer BEFORE you're able to use them. Do you remember the command to install a package?

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- 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 become errors
```

```
library(sf)
```

```
## Warning: package 'sf' was built under R version 4.4.1
```

```
## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```
library(terra)
```

```
## Warning: package 'terra' was built under R version 4.4.1
```

```
## terra 1.8.10
##
## Attaching package: 'terra'
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```r
library(tidyterra)
```

```
## Warning: package 'tidyterra' was built under R version 4.4.1
```

```
##
## Attaching package: 'tidyterra'
##
## The following object is masked from 'package:stats':
##
##     filter
```

```r
library(tmap)
```

```
## Warning: package 'tmap' was built under R version 4.4.1
```

A digression into geopackages

Let's load some data. Note, this is a different file format than you're (probably) used to. Check out https://www.geopackage.org if you want to learn more (and you should).

You may also notice the path is structured slightly differently that before. When in standard R script (for example, myscript.R), the "." notation refers to the location of the RStudio project file. HOWEVER, when using Rmd files, the starting location is where the .Rmd file is. Therefore, we need to edit our path a bit. ".." means "go up a level" (in this case, FROM the src directory and TO the root of the project) THEN find the `data` directory, then the `ohio` directory, then find the file.

Anyways, now we have some stream data. I like to always check the projection information. What's the projection?

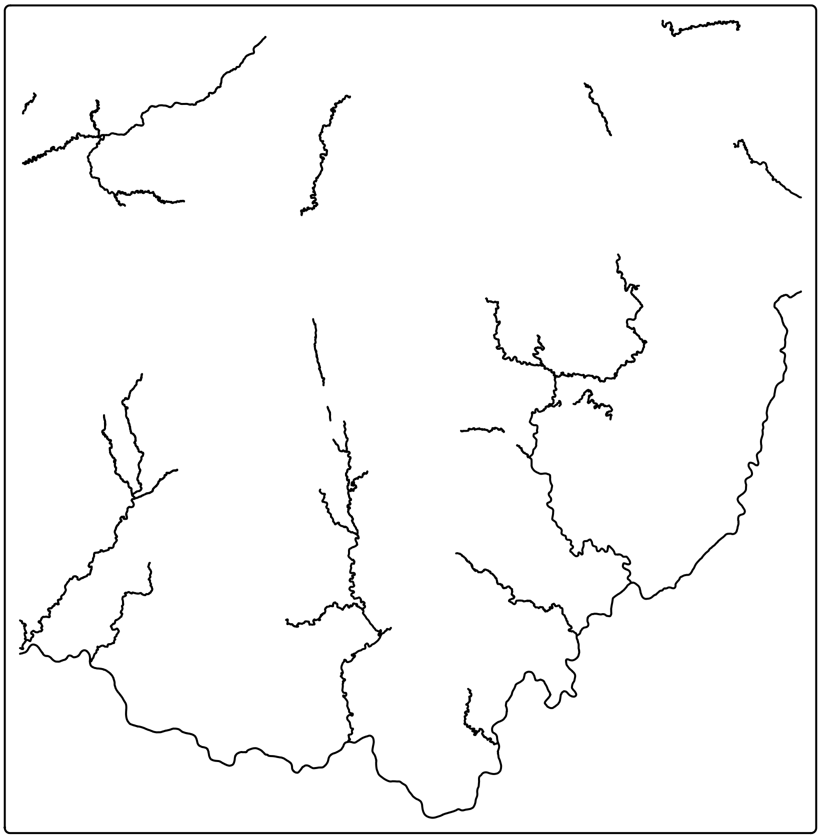
```r
oh_streams %>% sf::st_crs()
```

```
## Coordinate Reference System:
##   User input: WGS 84 / Pseudo-Mercator
##   wkt:
## PROJCRS["WGS 84 / Pseudo-Mercator",
##     BASEGEOGCRS["WGS 84",
##         ENSEMBLE["World Geodetic System 1984 ensemble",
##             MEMBER["World Geodetic System 1984 (Transit)"],
##             MEMBER["World Geodetic System 1984 (G730)"],
##             MEMBER["World Geodetic System 1984 (G873)"],
##             MEMBER["World Geodetic System 1984 (G1150)"],
##             MEMBER["World Geodetic System 1984 (G1674)"],
##             MEMBER["World Geodetic System 1984 (G1762)"],
##             MEMBER["World Geodetic System 1984 (G2139)"],
##             ELLIPSOID["WGS 84",6378137,298.257223563,
##                 LENGTHUNIT["metre",1]],
##             ENSEMBLEACCURACY[2.0]],
##         PRIMEM["Greenwich",0,
##             ANGLEUNIT["degree",0.0174532925199433]],
##         ID["EPSG",4326]],
##     CONVERSION["Popular Visualisation Pseudo-Mercator",
```

```
##         METHOD["Popular Visualisation Pseudo Mercator",
##             ID["EPSG",1024]],
##         PARAMETER["Latitude of natural origin",0,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8801]],
##         PARAMETER["Longitude of natural origin",0,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8802]],
##         PARAMETER["False easting",0,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8806]],
##         PARAMETER["False northing",0,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8807]]],
##     CS[Cartesian,2],
##         AXIS["easting (X)",east,
##             ORDER[1],
##             LENGTHUNIT["metre",1]],
##         AXIS["northing (Y)",north,
##             ORDER[2],
##             LENGTHUNIT["metre",1]],
##     USAGE[
##         SCOPE["Web mapping and visualisation."],
##         AREA["World between 85.06°S and 85.06°N."],
##         BBOX[-85.06,-180,85.06,180]],
##     ID["EPSG",3857]]
```

And then we can map it. I'm introducing a new package `tmap` today. This package does thematic mapping (hence, tmap) with various spatial data. The syntax uses the `+` notation similar to (but not exactly like) `ggplot`. You'll notice it's MUCH faster than the standard `plot()` command.

```
tm_shape(oh_streams) + tm_lines()
```



Let’s grab some more data

```r
oh_counties <- read_sf("../data/ohio/oh_counties.gpkg")
oh_counties %>% glimpse()
```

```
## Rows: 88
## Columns: 19
## $ STATEFP  <chr> "39", "39", "39", "39", "39", "39", "39", "39", "39", "39", "~
## $ COUNTYFP <chr> "063", "003", "085", "047", "017", "115", "133", "145", "163"~
## $ COUNTYNS <chr> "01074044", "01074015", "01074055", "01074036", "01074021", "~
## $ GEOID    <chr> "39063", "39003", "39085", "39047", "39017", "39115", "39133"~
## $ GEOIDFQ  <chr> "0500000US39063", "0500000US39003", "0500000US39085", "050000~
## $ NAME     <chr> "Hancock", "Allen", "Lake", "Fayette", "Butler", "Morgan", "P~
## $ NAMELSAD <chr> "Hancock County", "Allen County", "Lake County", "Fayette Cou~
## $ LSAD     <chr> "06", "06", "06", "06", "06", "06", "06", "06", "06", "06", "~
## $ CLASSFP  <chr> "H1", "H1", "H1", "H1", "H1", "H1", "H1", "H1", "H1", "H1", "~
## $ MTFCC    <chr> "G4020", "G4020", "G4020", "G4020", "G4020", "G4020", "G4020"~
## $ CSAFP    <chr> "248", "338", "184", "198", "178", NA, "184", "170", NA, NA, ~
## $ CBSAFP   <chr> "22300", "30620", "17410", "47920", "17140", NA, "10420", "39~
## $ METDIVFP <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ FUNCSTAT <chr> "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "~
## $ ALAND    <dbl> 1376122055, 1042587391, 594129618, 1052469885, 1208270096, 10~
## $ AWATER   <dbl> 6024245, 11152061, 1942308103, 1694038, 9196537, 13868572, 43~
```

```
## $ INTPTLAT <chr> "+41.0002170", "+40.7716274", "+41.7781416", "+39.5552462", "~
## $ INTPTLON <chr> "-083.6659471", "-084.1061032", "-081.1973297", "-083.4618927~
## $ geom     <MULTIPOLYGON [°]> MULTIPOLYGON (((-83.61191 4..., MULTIPOLYGON (((~
```

So now we have all counties in Ohio. Cool. Let's do some simple calculations with the data

```r
counties_areas <- oh_counties %>% sf::st_area()
```

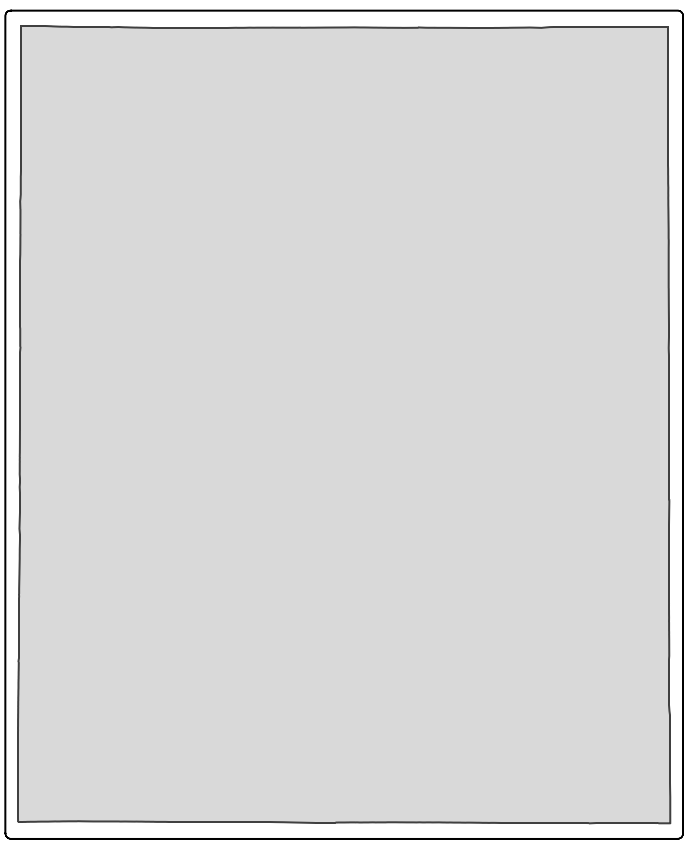
If you wanted to, how would you add the areas back to the sf data.frame?

Let's subset our data so that we're not working with ALL of Ohio. There are lots of ways to do this. How would we get ONLY Portage county?

```r
portage <- oh_counties %>% dplyr::filter(., NAME == "Portage")
```

Check it/plot it

```r
portage %>% tm_shape(.) + tm_polygons()
```



Yep, it's a rectangle.

Let's make a slightly larger study area to include Summit County as well. How could we do that? Let's just use an "or" within the filter command.

```
port.summit <- oh_counties %>% dplyr::filter(., NAME == "Portage" | NAME == "Summit")
```

As you can imagines, that can get a bit clunky if we need to string together a bunch of "or" commands. So let's try a different notation that's also a bit more reuseable.

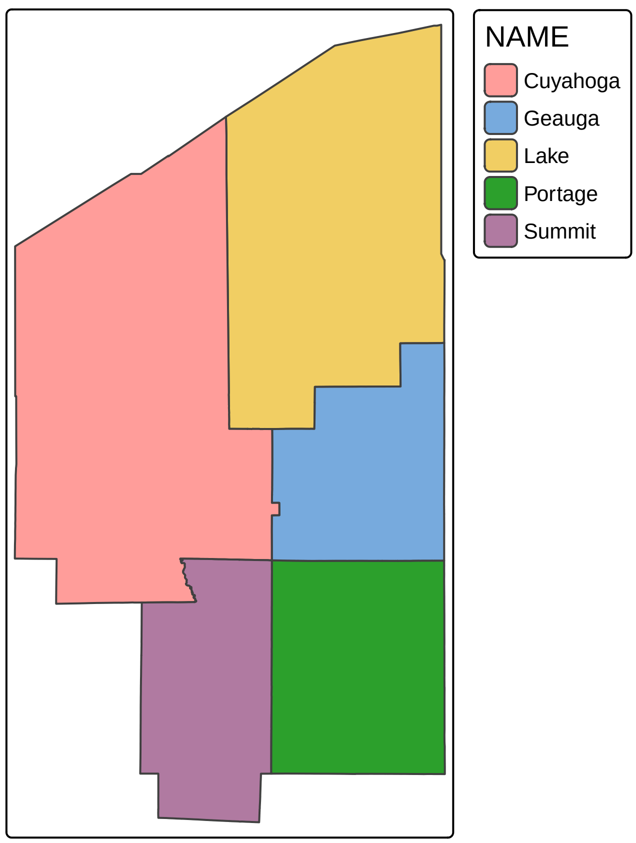
```
# what counties do I want?

# Make a simple vector
mycounties <- c("Portage", "Summit", "Lake", "Cuyahoga", "Geauga")

# then do the filter. Note the %in% notation. How do you think this works???
study.area <- oh_counties %>% dplyr::filter(., NAME %in% mycounties)
```

Plot it to check, add a fill based on a variable. It very handily adds a simple legend too!

```
study.area %>% tm_shape(.) + tm_polygons(fill = "NAME")
```



The streams dataset includes a variable for whether that stream segment is classified as impaired and on the "303d" list, which is list of impaired streams as defined by section 303d of the Clean Water Act. Let's filter the line file such that we only have those streams

```
streams.303d <- oh_streams %>% dplyr::filter(., on303dlist == "Y")
# It would make more sense if they used a logical (T/F) rather than Y/N, but I didn't create the data
```

Next, let's find only those 303d streams that are in our study area? What's the spatial operation again? Yes, an intersection

Oops, that didn't work. What was the problem?

Let's try again, this time dealing with the spatial reference/coordinate systems properly.

```
st_crs(study.area)
```

```
## Coordinate Reference System:
##   User input: NAD83
##   wkt:
## GEOGCRS["NAD83",
##     DATUM["North American Datum 1983",
##         ELLIPSOID["GRS 1980",6378137,298.257222101,
##             LENGTHUNIT["metre",1]]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433]],
##     CS[ellipsoidal,2],
##         AXIS["geodetic latitude (Lat)",north,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433]],
##         AXIS["geodetic longitude (Lon)",east,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433]],
##     USAGE[
##         SCOPE["Geodesy."],
##         AREA["North America - onshore and offshore: Canada - Alberta; British Columbia; Manitoba; Nev
##         BBOX[14.92,167.65,86.45,-40.73]],
##     ID["EPSG",4269]]
```

```
st_crs(oh_streams)
```

```
## Coordinate Reference System:
##   User input: WGS 84 / Pseudo-Mercator
##   wkt:
## PROJCRS["WGS 84 / Pseudo-Mercator",
##     BASEGEOGCRS["WGS 84",
##         ENSEMBLE["World Geodetic System 1984 ensemble",
##             MEMBER["World Geodetic System 1984 (Transit)"],
##             MEMBER["World Geodetic System 1984 (G730)"],
##             MEMBER["World Geodetic System 1984 (G873)"],
##             MEMBER["World Geodetic System 1984 (G1150)"],
##             MEMBER["World Geodetic System 1984 (G1674)"],
##             MEMBER["World Geodetic System 1984 (G1762)"],
##             MEMBER["World Geodetic System 1984 (G2139)"],
##             ELLIPSOID["WGS 84",6378137,298.257223563,
##                 LENGTHUNIT["metre",1]],
##             ENSEMBLEACCURACY[2.0]],
##         PRIMEM["Greenwich",0,
##             ANGLEUNIT["degree",0.0174532925199433]],
##         ID["EPSG",4326]],
##     CONVERSION["Popular Visualisation Pseudo-Mercator",
##         METHOD["Popular Visualisation Pseudo Mercator",
```

```
##             ID["EPSG",1024]],
##         PARAMETER["Latitude of natural origin",0,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8801]],
##         PARAMETER["Longitude of natural origin",0,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8802]],
##         PARAMETER["False easting",0,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8806]],
##         PARAMETER["False northing",0,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8807]]],
##     CS[Cartesian,2],
##         AXIS["easting (X)",east,
##             ORDER[1],
##             LENGTHUNIT["metre",1]],
##         AXIS["northing (Y)",north,
##             ORDER[2],
##             LENGTHUNIT["metre",1]],
##     USAGE[
##         SCOPE["Web mapping and visualisation."],
##         AREA["World between 85.06°S and 85.06°N."],
##         BBOX[-85.06,-180,85.06,180]],
##     ID["EPSG",3857]]
```

```r
# they're not the same, so we need to reproject them into a common CRS...


# The 6346 is an EPSG code (see: https://epsg.io) for a UTM 16N CRS

# let's reproject this one first
# ... or in `sf` parlance, "transform" it
study.area_p <- sf::st_transform(study.area, 6346)

# we COULD (and maybe should) use a similar command to reproject the streams file too.
#But let's do something a bit different/crazy just to show what's possible


# Before you run this next line, break down what it does FIRST. It's definitely non-traditional

study.area_p %>% st_crs() %>% sf::st_transform(study.area, .) -> oh_streams_p

# Now, while the above line technically works, it's not very readable,
# and an example of "just because you can, doesn't mean you should"

# something like this is probably better

oh_streams_p <- study.area_p %>% st_crs() %>% sf::st_transform(oh_streams, .)
```

Let's compare how the CRS impacts calcultions

```r
# unprojected areas
areas.unproj <- study.area %>% sf::st_area()

# projected areas
areas.proj <- study.area_p %>% sf::st_area()

# note that they're both in meters
# test for equality
areas.unproj == areas.proj
```

```
## [1] FALSE FALSE FALSE FALSE FALSE
```

```r
# test of differences
areas.unproj - areas.proj
```

```
## Units: [m^2]
## [1] -1812456.1  -701475.4  -648424.3 -2335643.9  -619086.6
```

```r
# differences as a percent of original
100 * (areas.unproj - areas.proj) / areas.unproj
```

```
## Units: [1]
## [1] -0.07156415 -0.05380353 -0.06138508 -0.07250693 -0.05696929
```

```r
# not a MASSIVE difference, but can still introduce error
```

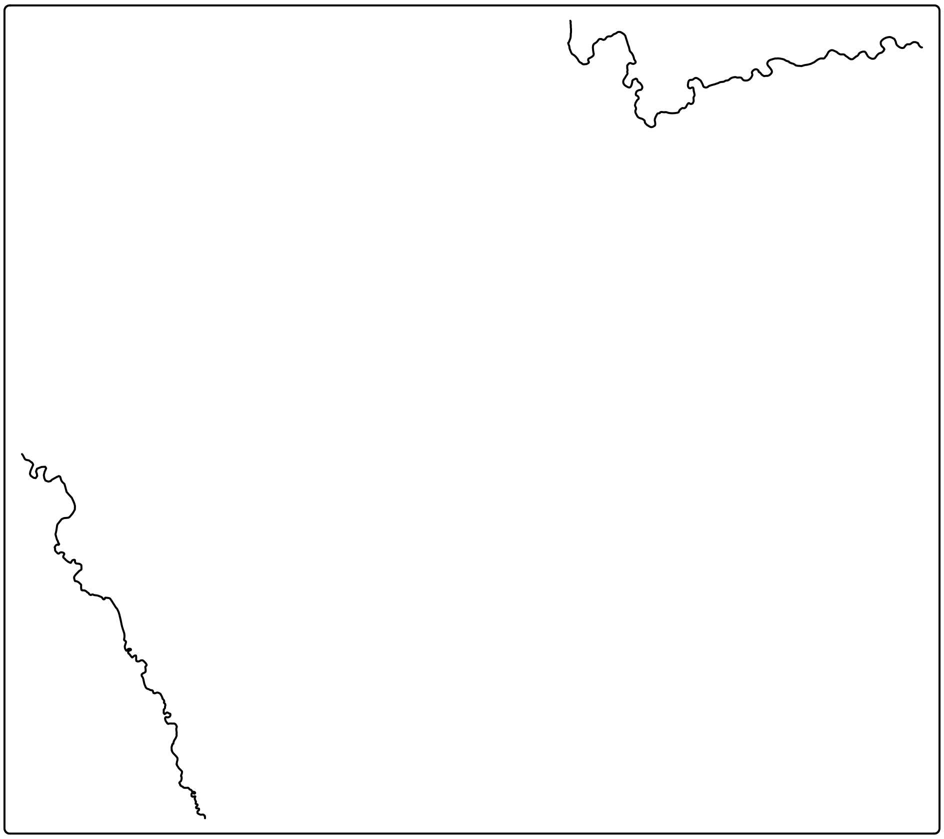
Now let's try that intersect function again

```r
study.streams <- sf::st_intersection(oh_streams_p, study.area_p)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout
## all geometries
```

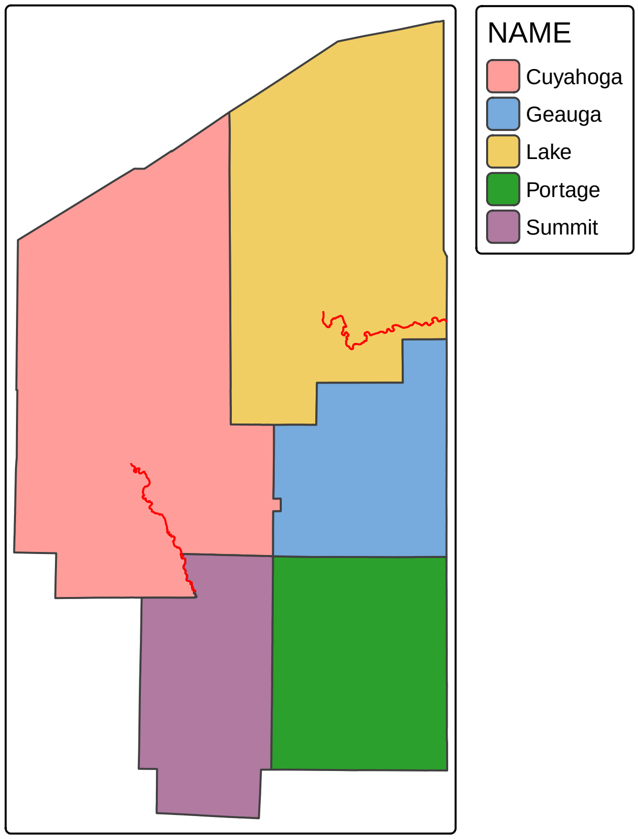
Plot it

```r
tm_shape(study.streams) + tm_lines()
```



Let's add both layers

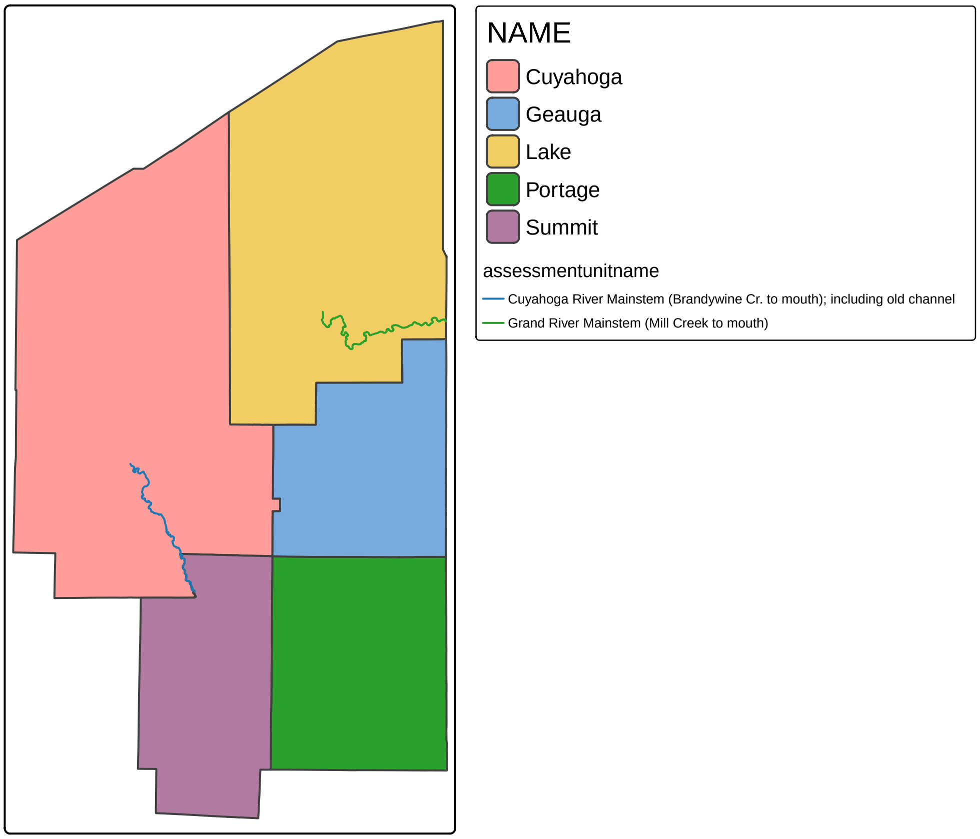
```
tm_shape(study.area_p) + tm_polygons(fill = "NAME") +
  tm_shape(study.streams) + tm_lines(col = "red") # this colors the lines based on a color we gave it (
```



Another option

```
tm_shape(study.area_p) + tm_polygons(fill = "NAME") +
  tm_shape(study.streams) + tm_lines(col = "assessmentunitname") # this colors the lines based on a var
```

```
## [plot mode] fit legend/component: Some legend items or map compoments do not
## fit well, and are therefore rescaled.
## i Set the tmap option ‘component.autoscale = FALSE‘ to disable rescaling.
```



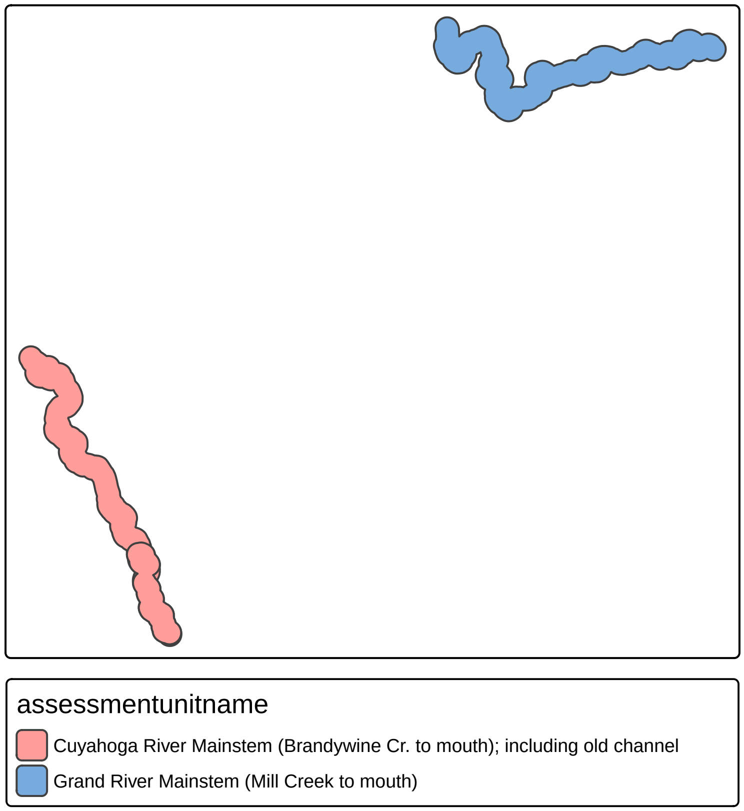
Buffers

What's a buffer?

Break down this code

```
buffs <- sf::st_buffer(study.streams, dist = 1000)

tm_shape(buffs) + tm_polygons(fill = "assessmentunitname")
```



Let's add some parks. There are two parks files in the `/data/ohio/` directory. One is a shapefile, one is a geopackage. What's the difference?

```
oh_parks_shp <- read_sf("../data/ohio/ohio_parks.shp")
oh_parks <- read_sf("../data/ohio/oh_parks.gpkg")
```

They're VERY similar, but there are some cases where they might not be the same. Think about when/where, and let's have a class discussion if you're not sure

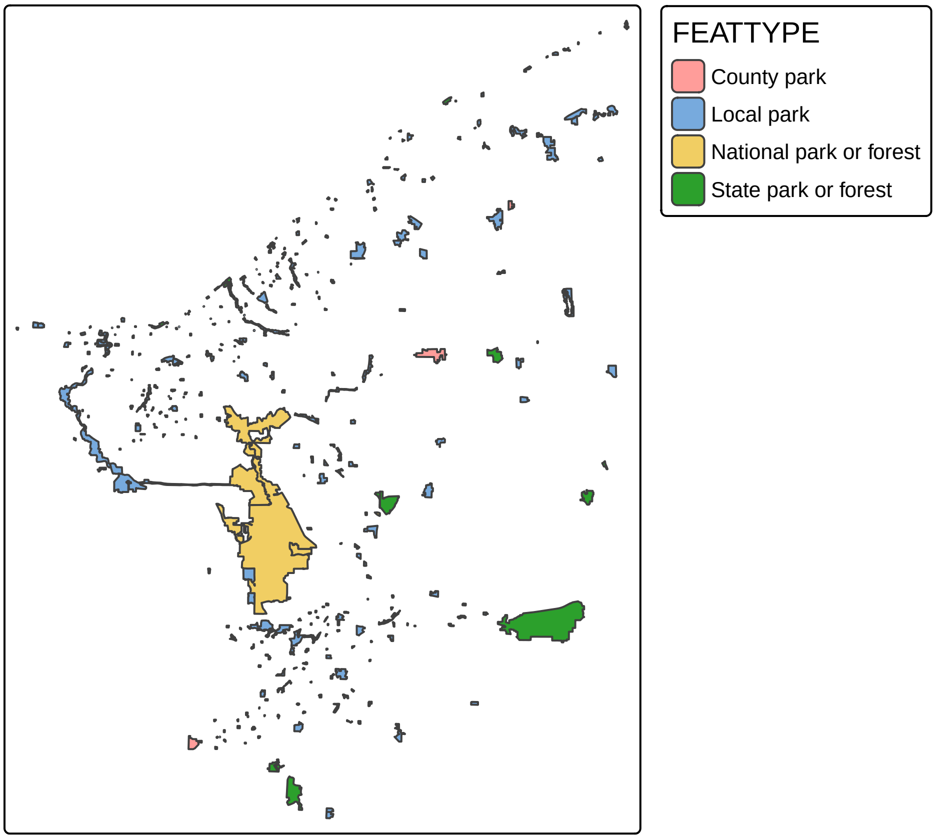
Let's subset the parks to our study area. Don't forget - we need to reproject first!

```
oh_parks_p <-  sf::st_transform(oh_parks, 6346)

oh_parks_p_studyarea <- sf::st_intersection(oh_parks_p, study.area_p)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout
## all geometries
```

```
tm_shape(oh_parks_p_studyarea) + tm_polygons(fill = "FEATTYPE")
```



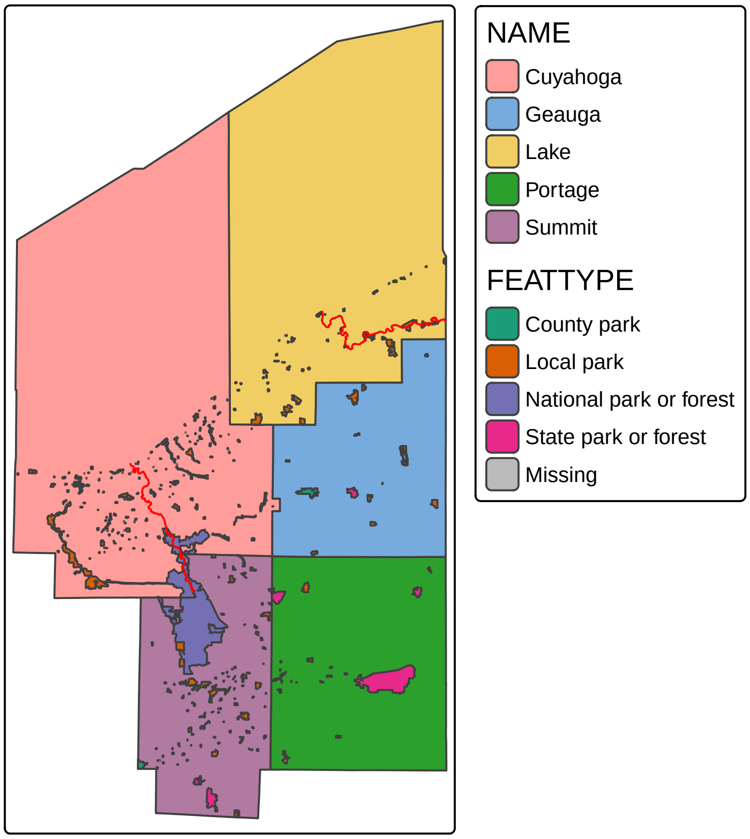
Let's do some more layering in a map - we can even change the palette we want to use!

```
tm_shape(study.area_p) + tm_polygons(fill = "NAME") +
  tm_shape(oh_parks_p_studyarea) + tm_polygons(fill = "FEATTYPE", palette = "brewer.dark2") +
  tm_shape(study.streams) + tm_lines(col = "red")
```

```
##

## -- tmap v3 code detected -------------------------------------------------------

## [v3->v4] ‘tm_tm_polygons()‘: migrate the argument(s) related to the scale of
## the visual variable ‘fill‘ namely ’palette’ (rename to ’values’) to fill.scale
## = tm_scale(<HERE>).
```




```
## Perhaps a bit on the ugly side, but it gets the point across
```

Distances

As an example, let's say we want to know the distance between parks and streams. We can use the following function

```r
sf::st_distance(study.streams, oh_parks_p_studyarea)
```

```
## Units: [m]
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
## [1,] 22903.89 19985.48 19581.43 18930.98 13271.50 15006.48 15850.68 20409.01
## [2,] 20469.59 23236.32 23853.66 24477.60 30864.93 28396.20 27740.20 21957.14
## [3,] 28803.70 32438.91 33623.14 33749.70 35383.65 35766.52 34577.88 27626.43
##          [,9]    [,10]    [,11]    [,12]    [,13]    [,14]    [,15]     [,16]
## [1,] 15269.61 15674.51 14903.85 13834.91 14269.67 10805.60 10147.83  9998.173
## [2,] 27941.75 30375.45 28379.45 29063.29 28912.50 32010.11 33035.20 33002.735
## [3,] 32190.51 33120.61 36340.01 37256.19 36693.77 40102.93 40592.34 39886.691
##         [,17]     [,18]     [,19]    [,20]     [,21]     [,22]     [,23]
## [1,] 10289.05  5925.886  8691.542  7082.18  7202.813  9007.164  7964.386
## [2,] 32961.63 36801.729 35219.157 36759.64 36204.864 35088.495 36318.158
## [3,] 41249.38 44758.510 43947.339 45341.68 44347.671 41043.249 41982.535
##          [,24]     [,25]       [,26]        [,27]     [,28]     [,29]
## [1,]  381.7103  2317.902    91.74687 6.172526e-01  360.4985  1019.516
```

```
## [2,] 44881.7588 42119.390 43529.79683 4.527377e+04 44693.0189 47929.125
## [3,] 52284.6313 48336.898 51242.63223 5.122627e+04 50689.7497 54662.347
##         [,30]    [,31]     [,32]     [,33]     [,34]     [,35]     [,36]
## [1,]     0.00  6828.66  7019.289  6956.902  7770.591  8038.404  8215.733
## [2,] 44182.32 54342.28 54748.757 54905.911 57789.287 58665.898 60171.130
## [3,] 49264.96 60531.26 60943.178 61061.146 63765.401 64548.748 65918.810
##          [,37]    [,38]      [,39]     [,40]    [,41]    [,42]    [,43]
## [1,]  8917.643     0.00   730.9189  1320.914     0.00     0.00     0.00
## [2,] 62147.894 48263.43 51018.2108 50790.195 55330.77 48900.57 58605.66
## [3,] 67704.179 52923.93 54998.3337 53806.729 59170.21 53521.21 61832.60
##             [,44]       [,45]    [,46]     [,47]    [,48]    [,49]    [,50]
## [1,]     3.910866    10.68891 10200.49  9555.622 72366.39 49791.75 46962.28
## [2,] 59758.682287 60551.17759 68421.28 65231.281 37820.96 14647.03 14514.87
## [3,] 62949.271115 63588.90409 73377.60 70569.733 37936.43 14686.65 14536.13
##         [,51]    [,52]    [,53]    [,54]    [,55]    [,56]    [,57]    [,58]
## [1,] 59763.37 62849.75 61331.97 63211.26 63227.11 63184.84 63132.37 61556.41
## [2,] 20661.16 21880.69 21601.15 23126.91 22125.64 24706.53 25488.65 22719.68
## [3,] 20765.35 22001.33 21711.42 23242.35 22243.75 24817.79 25597.77 22829.88
##         [,59]    [,60]    [,61]    [,62]    [,63]    [,64]    [,65]    [,66]
## [1,] 61972.26 64957.09 62989.65 59544.82 58237.33 61145.75 60591.47 62306.69
## [2,] 22423.06 24011.99 24580.49 29024.33 24558.46 29345.22 31961.81 28852.33
## [3,] 22534.52 24133.55 24691.13 29109.33 24646.58 29437.22 32045.36 28949.93
##         [,67]    [,68]    [,69]    [,70]    [,71]    [,72]    [,73]    [,74]
## [1,] 44391.83 46190.14 60176.39 43313.18 76211.79 23732.02 20401.05 25676.35
## [2,] 20063.39 40820.14 34206.28 43894.37 34141.01 25208.61 44474.75 39610.55
## [3,] 20083.79 40842.69 34290.50 43902.82 34276.55 26117.13 44473.88 39610.39
##         [,75]    [,76]    [,77]    [,78]     [,79]    [,80]     [,81]    [,82]
## [1,] 15762.04 16541.75 32541.90 27435.78  7654.877 26765.11  6931.545 25863.84
## [2,] 38760.01 30645.31 35181.47 36001.30 40373.314 24229.76 43412.435 32926.94
## [3,] 39171.23 33153.14 35181.29 36001.17 42677.766 24281.99 45608.774 32926.78
##         [,83]    [,84]     [,85]      [,86]    [,87]    [,88]    [,89]    [,90]
## [1,] 38278.45 32012.02  5339.897 46871.8512 65272.66 60942.09 63534.20 60330.78
## [2,] 22973.51 47139.95 56816.922   661.2607 18464.92 18357.22 20990.36 17576.55
## [3,] 22973.38 47139.78 58808.252 13801.3579 22673.95 31541.73 33747.99 30888.64
##         [,91]    [,92]    [,93]    [,94]    [,95]    [,96]     [,97]     [,98]
## [1,] 57850.71 62140.34 62025.74 53988.05 55754.19 50765.60 51915.731 52860.180
## [2,] 14343.53 15867.79 15488.46  9493.54 10638.25  7028.38  7858.069  8565.634
## [3,] 26785.77 22146.12 21242.42 16910.33 20228.29 22865.42 22858.190 22552.509
##         [,99]    [,100]   [,101]   [,102]   [,103]    [,104]    [,105]
## [1,] 56949.08 52731.919 55061.68 56557.91 55701.63 53197.853 50950.503
## [2,] 12546.86  8742.134 11510.86 12418.52 11426.21  8629.638  6387.126
## [3,] 23933.88 23442.480 25723.87 24929.71 23940.39 21326.844 20479.851
##         [,106]     [,107]     [,108]    [,109]    [,110]    [,111]    [,112]
## [1,] 50035.929 45833.2534 44811.3881 52274.978 51661.086 48907.222 46332.747
## [2,]  5689.157   784.0998   732.3063  7317.178  6622.067  4146.947  1646.025
## [3,] 21282.248 17708.5633 16072.0510 19171.448 18385.839 19171.226 18985.950
##        [,113]    [,114]    [,115]    [,116]    [,117]    [,118]    [,119]
## [1,] 54786.37 53321.095 56815.698 54279.754 54618.683 52771.470 50997.998
## [2,]  7864.00  6514.189  9851.732  7453.883  8451.992  5805.052  4657.963
## [3,] 15156.41 14267.071 15940.420 15317.373 17414.285 14014.780 15189.840
##         [,120]    [,121]    [,122]    [,123]    [,124]    [,125]    [,126]
## [1,] 49891.352 54634.727 55617.551 54967.417 56116.672 55939.747 40004.224
## [2,]  1430.270  7166.534  8559.805  7690.758  8756.467  8742.345  3693.731
## [3,]  4779.552 12109.284 14354.086 13301.790 13623.814 14192.955 19790.984
```

```
##         [,127]    [,128]    [,129]    [,130]    [,131]    [,132]    [,133]
## [1,] 41249.582 36584.428 39456.595 36300.599 37980.045 34605.690 35242.157
## [2,]  2607.127  6300.436  3878.925  6158.612  5197.112  8348.094  8299.631
## [3,] 18909.465 20365.017 18242.561 20794.929 20575.443 21159.099 21510.495
##        [,134]    [,135]    [,136]   [,137]    [,138]    [,139]    [,140]
## [1,] 35138.60 37134.518 43785.822 43359.39 36755.486 52759.874 53073.671
## [2,]  8183.72  5668.696  9899.998     0.00  6775.027  5157.723  5539.813
## [3,] 19607.75 17901.902  9899.965     0.00 17967.311 10336.234  8782.167
##         [,141]    [,142]    [,143]    [,144]   [,145]   [,146]    [,147]
## [1,] 52472.857 52473.751 54136.663 50479.549 54440.87 51320.61 53611.447
## [2,]  5092.331  5190.697  6510.378  2968.894  6832.60  4116.99  6062.089
## [3,] 11743.405 12124.562  9803.738 10473.676 11229.05 11916.62 11309.950
##         [,148]    [,149]    [,150]    [,151]   [,152]    [,153]    [,154]
## [1,] 51144.254 49530.847 49103.323 51232.615 48180.39 48211.312 49533.953
## [2,]  3601.706  2502.094  1484.986  3557.790  1538.03  1549.542  3157.337
## [3,] 10413.451 11886.952  8806.879  7033.403 13639.88 13683.000 14566.021
##          [,155]    [,156]     [,157]     [,158]    [,159]     [,160]    [,161]
## [1,] 45445.7906 46738.387 43772.7776 43439.4571 43137.476 44641.7933 43327.316
## [2,]   402.5434  1944.005   643.1127   776.8823  1110.087   945.9312  2204.715
## [3,] 17795.4163 18669.583 18853.0588 18679.6262 18938.509 11857.3474 12728.447
##         [,162]    [,163]    [,164]    [,165]    [,166]    [,167]   [,168]
## [1,] 42728.824 41542.793 41513.654 40684.338 39973.390 40516.261 41559.76
## [2,]  2177.994  1979.459  6543.739  6167.759  3851.425  6112.582  1901.70
## [3,] 13390.409 17106.768  7838.388  8873.712 16141.708 10352.364 16216.06
##         [,169]    [,170]    [,171]    [,172]    [,173]    [,174]   [,175]
## [1,] 38438.029 37946.810 37552.135 35496.291 37468.884 34721.454 40278.47
## [2,]  6622.692  6070.182  5852.697  7354.821  5744.508  9497.565  3963.15
## [3,] 14775.399 16772.275 17194.758 14274.814 15563.692 17468.431 13275.05
##        [,176]   [,177]    [,178]   [,179]    [,180]    [,181]    [,182]
## [1,] 34164.68 35798.30 40828.722 41692.58 41931.777 42360.756 33995.268
## [2,] 12129.03 13409.79  4335.488  4002.37  6700.258  4348.769  8098.672
## [3,] 16646.30 13886.10 13050.212 12857.95  6795.156 11328.302 18524.087
##        [,183]    [,184]   [,185]    [,186]    [,187]    [,188]   [,189]
## [1,] 32675.81 39978.786 33260.66 43211.029 34867.077 43427.146 31993.36
## [2,] 10890.82  5088.626 10268.34  3605.262  9177.541  2529.711 11604.86
## [3,] 20773.70 13308.621 21079.94 10947.093 17706.821  8196.221 24080.08
##        [,190]   [,191]   [,192]   [,193]   [,194]   [,195]   [,196]   [,197]
## [1,] 32721.26 31463.75 32800.18 32873.46 28473.37 28473.21 64978.79 28482.58
## [2,] 11221.16 11964.84 10732.16 10533.75 14025.22 14998.00 17409.17 14599.39
## [3,] 19163.18 22662.82 22822.51 22163.28 21506.39 24751.01 18983.68 25116.08
##        [,198]   [,199]   [,200]   [,201]   [,202]   [,203]   [,204]   [,205]
## [1,] 29031.04 61840.67 30432.57 31193.28 30062.13 30686.03 32298.04 32080.94
## [2,] 14206.12 14329.79 13025.97 12121.11 13279.33 13363.97 12357.53 11604.55
## [3,] 24585.15 16795.53 23391.96 21880.00 23826.89 21337.28 18978.63 21427.68
##        [,206]   [,207]   [,208]   [,209]   [,210]   [,211]   [,212]   [,213]
## [1,] 30923.85 25109.91 28243.01 29768.51 28154.19 28885.49 59087.01 29640.46
## [2,] 13115.66 18635.83 15389.66 13422.48 15196.80 14441.18 11480.63 13861.28
## [3,] 20936.88 29645.81 27044.08 25126.58 25854.36 26170.02 14091.48 25644.32
##        [,214]   [,215]   [,216]    [,217]   [,218]    [,219]   [,220]    [,221]
## [1,] 58234.59 29134.08 59774.82 56515.331 24324.90 56431.692 22857.02 56388.103
## [2,] 10646.56 14214.32 12226.82  8887.549 18873.25  8799.042 20532.79  8812.844
## [3,] 13660.80 25736.40 15007.20 12191.940 29620.56 11026.611 30048.57 12565.797
##        [,222]   [,223]   [,224]   [,225]    [,226]    [,227]    [,228]   [,229]
## [1,] 25381.03 57847.23 25408.17 57822.37 54787.073 55646.277 53844.259 27139.32
```

```
## [2,] 17664.22 10433.65 17503.89 10455.64  6657.242  3560.475  5384.585 17277.10
## [3,] 28006.52 14311.52 27019.07 14599.51  8579.440  3560.802  6846.727 24107.40
##         [,230]   [,231]   [,232]   [,233]   [,234]   [,235]   [,236]   [,237]
## [1,] 29934.01 29202.02 29480.17 27382.51 28326.12 26680.00 23483.49 21782.89
## [2,] 16299.75 15938.99 15970.33 15910.93 15732.34 18282.90 21582.53 22225.60
## [3,] 20023.96 21518.41 20969.12 25442.81 23242.38 23626.26 26898.00 27773.62
##         [,238]   [,239]   [,240]   [,241]   [,242]   [,243]   [,244]   [,245]
## [1,] 31136.40 39697.78 35797.83 67606.40 67968.95 64313.52 64320.90 71911.59
## [2,] 16665.63 13056.62 11162.58 16276.61 17990.11 12636.89 13454.38 19488.74
## [3,] 16717.84 13056.51 11217.71 16422.62 18136.61 12784.11 13601.32 19632.83
##          [,246]    [,247]    [,248]   [,249]   [,250]   [,251]   [,252]   [,253]
## [1,] 58507.221 58693.385 57381.880 65309.95 63751.89 63212.01 66383.18 68823.97
## [2,]  4817.834  8582.393  6629.597 16988.60 16042.74 15893.64 17508.40 21325.69
## [3,]  4817.874  8723.966  6769.826 17130.56 16181.39 16030.36 17652.40 21468.27
##         [,254]   [,255]   [,256]   [,257]   [,258]   [,259]   [,260]   [,261]
## [1,] 69654.13 67638.87 66366.58 65081.15 64848.63 67047.24 67101.64 65709.71
## [2,] 20577.34 17890.67 16501.11 16059.83 17538.29 19525.96 18897.64 18387.49
## [3,] 20723.22 18036.80 16646.93 16203.49 17676.83 19667.25 19043.49 18527.13
##          [,262]    [,263]    [,264]    [,265]    [,266]    [,267]    [,268]
## [1,] 46052.557 54891.302 46189.374 43787.217 47810.660 49153.735 46513.151
## [2,]  5052.649  6438.543  8228.509  9900.666  6569.919  5175.057  6794.957
## [3,]  5052.622  6544.768  8228.273  9900.633  6569.874  5173.052  6794.591
##         [,269]   [,270]   [,271]   [,272]   [,273]   [,274]   [,275]    [,276]
## [1,] 48938.47 50238.98 51469.04 49777.76 44795.39 44294.51 43929.80 46743.081
## [2,]     0.00 12767.19 13174.64 13997.11 11161.30 10116.46 10510.35  9718.052
## [3,]     0.00 12808.82 13237.80 14036.95 11161.13 10115.54 10509.95  9717.938
##          [,277]   [,278]   [,279]   [,280]   [,281]   [,282]   [,283]   [,284]
## [1,] 46199.971 47096.25 64722.78 64013.10 63433.61 61007.15 66235.30 64990.34
## [2,]  9322.758 13849.18 20478.52 17892.87 18041.66 18608.44 23157.85 18729.00
## [3,]  9322.584 13867.58 20608.59 18027.07 18173.03 18724.96 23286.62 18865.00
##         [,285]   [,286]   [,287]   [,288]   [,289]   [,290]   [,291]    [,292]
## [1,] 66030.40 58019.68 55130.86 55655.18 54662.39 56452.49 63795.33 63071.635
## [2,] 19664.11 16457.42 11159.98 13736.68 12842.38 15072.11 22014.90  9641.840
## [3,] 19801.73 16561.67 11254.28 13830.23 12927.49 15168.00 22136.00  9782.255
##         [,293]   [,294]   [,295]   [,296]   [,297]   [,298]   [,299]   [,300]
## [1,] 72236.49 71255.18 70792.73 66159.72 72093.85 70121.27 84875.01 86184.84
## [2,] 19020.66 17758.62 17629.68 12222.53 18756.76 17237.68 32431.04 33592.19
## [3,] 19159.12 17893.95 17769.30 12222.54 18894.04 17379.80 32572.32 33732.46
##         [,301]   [,302]   [,303]   [,304]   [,305]   [,306]   [,307]   [,308]
## [1,] 81698.56 80398.25 80778.21 82427.51 84930.96 83726.33 84782.28 84127.46
## [2,] 29504.52 28485.68 28923.66 30356.75 32245.61 31197.14 32171.80 32442.95
## [3,] 29647.48 28629.89 29068.00 30500.08 32385.51 31338.15 32312.18 32587.31
##         [,309]   [,310]   [,311]   [,312]   [,313]   [,314]   [,315]   [,316]
## [1,] 82994.85 86300.91 88843.22 89326.95 84657.25 81783.36 82651.64 79155.07
## [2,] 30109.36 33092.58 41603.24 42067.05 34509.99 30286.76 31484.73 27691.47
## [3,] 30248.26 33224.77 41750.06 42213.90 34656.88 30431.93 31630.50 27837.07
##         [,317]   [,318]   [,319]   [,320]   [,321]   [,322]   [,323]   [,324]
## [1,] 79137.47 79839.89 72363.07 77764.55 72456.15 74051.75 73225.07 74448.98
## [2,] 28656.01 28697.51 20741.00 26596.08 20930.06 22172.77 22700.74 24314.25
## [3,] 28802.95 28843.64 20887.31 26742.36 21076.54 22317.99 22847.99 24461.51
##         [,325]   [,326]   [,327]   [,328]   [,329]   [,330]   [,331]   [,332]
## [1,] 75627.01 76829.81 76856.59 73573.83 72492.11 72236.70 71416.25 69538.38
## [2,] 25182.71 25438.37 27390.28 24902.81 22723.32 22918.19 22269.71 22137.74
## [3,] 25329.82 25584.25 27537.46 25049.02 22870.36 23064.84 22416.05 22280.80
```

```
##         [,333]   [,334]   [,335]   [,336]   [,337]   [,338]   [,339]   [,340]
## [1,] 72040.35 71747.96 74808.67 79121.67 77599.81 76815.89 77103.22 75309.61
## [2,] 24882.71 24759.70 26852.91 29842.46 28556.12 31699.51 31070.93 28502.79
## [3,] 25026.28 24902.72 26998.68 29989.67 28703.17 31841.37 31214.34 28647.24
##         [,341]   [,342]   [,343]   [,344]   [,345]
## [1,] 73961.79 64617.35 68674.30 70707.07 73011.60
## [2,] 27725.30 16161.93 25155.33 24237.92 27484.12
## [3,] 27867.97 16303.60 25288.39 24380.19 27624.88
```

Yikes, that's a bit of a mess. Let's turn that into a tibble (which is a kind of fancy table, and also a replacement for the tidyverse data frame)

```
sf::st_distance(study.streams, oh_parks_p_studyarea) %>% as_tibble()
```

```
## # A tibble: 3 x 345
##       V1     V2     V3     V4     V5     V6     V7     V8     V9    V10    V11
##      [m]    [m]    [m]    [m]    [m]    [m]    [m]    [m]    [m]    [m]    [m]
## 1 22904. 19985. 19581. 18931. 13271. 15006. 15851. 20409. 15270. 15675. 14904.
## 2 20470. 23236. 23854. 24478. 30865. 28396. 27740. 21957. 27942. 30375. 28379.
## 3 28804. 32439. 33623. 33750. 35384. 35767. 34578. 27626. 32191. 33121. 36340.
## # i 334 more variables: V12 [m], V13 [m], V14 [m], V15 [m], V16 [m], V17 [m],
## #   V18 [m], V19 [m], V20 [m], V21 [m], V22 [m], V23 [m], V24 [m], V25 [m],
## #   V26 [m], V27 [m], V28 [m], V29 [m], V30 [m], V31 [m], V32 [m], V33 [m],
## #   V34 [m], V35 [m], V36 [m], V37 [m], V38 [m], V39 [m], V40 [m], V41 [m],
## #   V42 [m], V43 [m], V44 [m], V45 [m], V46 [m], V47 [m], V48 [m], V49 [m],
## #   V50 [m], V51 [m], V52 [m], V53 [m], V54 [m], V55 [m], V56 [m], V57 [m],
## #   V58 [m], V59 [m], V60 [m], V61 [m], V62 [m], V63 [m], V64 [m], V65 [m], ...
```

That's more intepretable. We have a 3 x 345 table. How might you infer what each row and each column represent? (Hint, go back to the data you gave to the `st_distance()` function)

YOUR TASKS

I have given you all of the tools to complete the following items:

Task(s) 2

- find ALL of the Ohio parks within 1km of a 303d stream
- Make a map (using `tmap`) of just those parks (not all parks), overlaid on a Ohio county map
- Add a color to the parks based on the type of park (like we did)

Task(s) 2

- Calculate the distance between parks and 303d streams in the study area
- Then, calculate the *difference* between projected and unprojected distances
- Then, make a histogram of those differences (better termed "errors")

Bonus work

- How many counties do NOT have a 303d stream in them?
- What county intersects the most parks? The most streams? The most 303d streams?