Geoprocessing 1 - Geometric Measurements

Code

In this Notebook we will use functions from `sf` that measure geometric properties (area, length, distance).

- compute the total area of YNP in meters^2 and miles^2
- compute the area of watersheds
- compute the total length of roads
- compute the distance between campgrounds and cell towers

Setup

Load the packages we'll need and set tmap mode to 'plot':

Hide

```
library(sf)
library(units)
library(tmap)
tmap_mode("plot")
```

```
tmap mode set to plotting
```

Load `dplyr` and set name conflict preferences:

Hide

```
library(dplyr)

## Load the conflicted package
library(conflicted)

# Set conflict preferences
conflict_prefer("filter", "dplyr", quiet = TRUE)
conflict_prefer("count", "dplyr", quiet = TRUE)
conflict_prefer("select", "dplyr", quiet = TRUE)
conflict_prefer("arrange", "dplyr", quiet = TRUE)
```

Import the Park Boundary

Hide

```
## Define a convenience variable for UTM Zone 11
epsg_utm11n_nad83 <- 26911

## Import the YNP border
yose_bnd_utm <- st_read(dsn="./data", layer="yose_boundary") |>
  st_transform(epsg_utm11n_nad83)
```

```
Reading layer `yose_boundary' from data source
  `D:\Workshops\R-Spatial\rspatial_mod\outputs\rspatial_bgs23\notebooks\data' using driver `ESRI Shapefile'
Simple feature collection with 1 feature and 11 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -119.8864 ymin: 37.4947 xmax: -119.1964 ymax: 38.18515
Geodetic CRS:  North_American_Datum_1983
```

Compute the area of the park

Find area with `st_area()`:

Hide

```
yose_area <- yose_bnd_utm |> st_area()
yose_area
```

```
3019918537 [m^2]
```

Find the area in miles^2.

Hide

```
## View in square miles
set_units(yose_area, mi^2)
```

```
1165.997 [mi^2]
```

Import the Watersheds

Next import the watershed boundaries.

Hide

```
yose_watersheds_utm <- st_read("./data/yose_watersheds.gpkg", layer="calw221") |>
  st_transform(epsg_utm11n_nad83) |>
  select(CALWNUM, HRNAME, RBNAME, HUNAME, CDFSPWNAME, CDFPWSNAME, HUC_8, HUC_8_NAME)
```

```
Reading layer `calw221' from data source
  `D:\Workshops\R-Spatial\rspatial_mod\outputs\rspatial_bgs23\notebooks\data\yose_watersheds.gpkg'
  using driver `GPKG'
Simple feature collection with 127 features and 12 fields
```

```
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: 1383.82 ymin: -61442.93 xmax: 81596.71 ymax: 26405.66
Projected CRS: NAD83 / California Albers
```

Hide

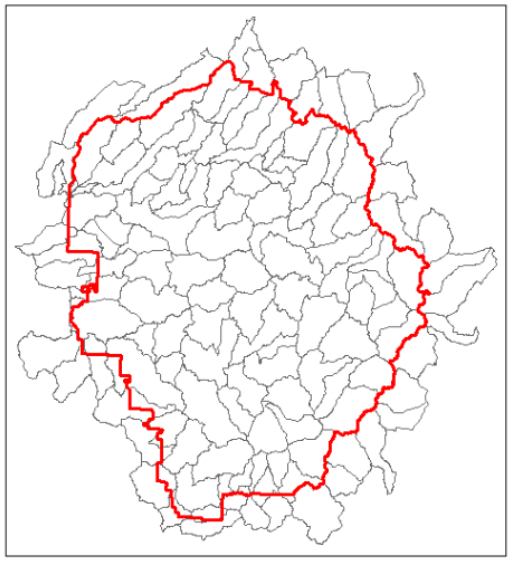
```
yose_watersheds_utm |> slice(1:10) |> as_tibble()
```

CALWNUM <chr>	HRNAME <chr>	RBNAME <chr>	HUNAME <chr>	CDFSPWNAME <chr>
8631.400101	North Lahontan	Lahontan	WEST WALKER RIVER	Headwaters West Walker River
8630.400204	North Lahontan	Lahontan	EAST WALKER RIVER	Buckeye Creek
8631.400104	North Lahontan	Lahontan	WEST WALKER RIVER	Headwaters West Walker River
6536.510101	San Joaquin	Central Valley	TUOLUMNE RIVER	Huckleberry Lake
6536.600801	San Joaquin	Central Valley	TUOLUMNE RIVER	Falls Creek
8630.400301	North Lahontan	Lahontan	EAST WALKER RIVER	Twin Lakes
8630.400302	North Lahontan	Lahontan	EAST WALKER RIVER	Twin Lakes
8630.400400	North Lahontan	Lahontan	EAST WALKER RIVER	Green Creek
8630.400306	North Lahontan	Lahontan	EAST WALKER RIVER	Twin Lakes
6536.600701	San Joaquin	Central Valley	TUOLUMNE RIVER	Rancheria Creek

1-10 of 10 rows | 1-5 of 9 columns

Hide

```
tm_shape(yose_watersheds_utm) +
  tm_borders() +
tm_shape(yose_bnd_utm) +
  tm_borders(col="red", lwd=3)
```



CHALLENGE: Compute the watershed areas

Compute the watershed areas, and add them as a new column in the attribute table. Answer

Hide

```
## Compute the area in map units
yose_watersheds_areas_m2 <- st_area(yose_watersheds_utm)

## Add the area to the attribute table
yose_watersheds_utm <- yose_watersheds_utm |>
  mutate(area_m2 = yose_watersheds_areas_m2)

## Preview results
yose_watersheds_utm |> slice(1:10) |> as_tibble()
```

CALWNUM <chr>	HRNAME <chr>	RBNAME <chr>	HUNAME <chr>	CDFSPWNAME <chr>
8631.400101	North Lahontan	Lahontan	WEST WALKER RIVER	Headwaters West Walker River
8630.400204	North Lahontan	Lahontan	EAST WALKER RIVER	Buckeye Creek
8631.400104	North Lahontan	Lahontan	WEST WALKER RIVER	Headwaters West Walker River
6536.510101	San Joaquin	Central Valley	TUOLUMNE RIVER	Huckleberry Lake
6536.600801	San Joaquin	Central Valley	TUOLUMNE RIVER	Falls Creek
8630.400301	North Lahontan	Lahontan	EAST WALKER RIVER	Twin Lakes

8630.400302	North Lahontan	Lahontan	EAST WALKER RIVER	Twin Lakes
8630.400400	North Lahontan	Lahontan	EAST WALKER RIVER	Green Creek
8630.400306	North Lahontan	Lahontan	EAST WALKER RIVER	Twin Lakes
6536.600701	San Joaquin	Central Valley	TUOLUMNE RIVER	Rancheria Creek

1-10 of 10 rows | 1-5 of 10 columns

CHALLENGE: Find the largest watershed

Find the largest watershed, and report the area in acres. Answer

Hide

```
## Find the biggest watershed
yose_watersheds_utm |>
  st_drop_geometry() |>     ## get rid of the geometry column - don't need it
  top_n(1, area_m2)
```

CALWNUM <chr>	HRNAME <chr>	RBNAME <chr>	HUNAME <chr>	CDFSPWN... <chr>	CDFPWSNAME <chr>	HUC_8 <dbl>
6537.500304	San Joaquin	Central Valley	MERCED RIVER	McCauley	Crane Creek	18040008

1 row | 1-7 of 9 columns

Hide

```
## Compute the area in acres
yose_watersheds_utm |>
  st_drop_geometry() |>     ## get rid of the geometry column - don't need it
  top_n(1, area_m2) |>
  pull(area_m2) |>
  set_units("acres")
```

```
20017.12 [acres]
```

Import the Roads

Hide

```
yose_roads_utm <- st_read("./data/yose_roads.gdb", "Yosemite_Roads") |>
  st_transform(epsg_utm11n_nad83) |>
  select(RDNAME, RTENUMBER, YOSE_Surface, YOSE_FIRE_ROAD, YOSE_INPARK, YOSE_Type)
```

```
Reading layer `Yosemite_Roads' from data source
  `D:\Workshops\R-Spatial\rspatial_mod\outputs\rspatial_bgs23\notebooks\data\yose_roads.gdb'
  using driver `OpenFileGDB'
Simple feature collection with 823 features and 40 fields
Geometry type: MULTILINESTRING
Dimension:     XY
Bounding box:  xmin: 234658.1 ymin: 4139484 xmax: 324852.6 ymax: 4250252
Projected CRS: NAD83 / UTM zone 11N
```

Compute the length of roads and save it in the attribute table:

Hide

```
yose_roads_utm <- yose_roads_utm |>
  mutate(rd_length = st_length(yose_roads_utm))
yose_roads_utm |> slice(1:10) |> as_tibble()
```

RDNAME <chr>	RTENUMB... <chr>	YOSE_Surface <chr>	YOSE_FIRE_ROAD <chr>	YOSE_INPARK <chr>	YOSE_Type <chr>
Northside Drive	*NA*	Paved	No	Yes	Primary Road
Northside Drive	*NA*	Paved	No	Yes	Primary Road
Northside Drive	*NA*	Paved	No	Yes	Primary Road
Southside Drive	*NA*	Paved	No	Yes	Primary Road
Northside Drive	*NA*	Paved	No	Yes	Primary Road
Northside Drive	*NA*	Paved	No	Yes	Primary Road
Mirror Lake road	*NA*	Paved	No	Yes	Secondary Road
Happy Isles Loop	*NA*	Paved	No	Yes	Primary Road
Mirror Lake road	*NA*	Paved	No	Yes	Secondary Road
Happy Isles Loop	*NA*	Paved	No	Yes	Primary Road

1-10 of 10 rows | 1-6 of 8 columns

CHALLENGE: Compute Road Lengths

Find the total length for each type of Road (see YOSE_Type) in miles. Answer

Hide

```
yose_roads_utm |>
  st_drop_geometry() |>
  group_by(YOSE_Type) |>
```

```
  summarise(length_miles = sum(rd_length) |> set_units("mi")) |>
  arrange(desc(length_miles))
```

YOSE_Type <chr>	length_miles <S3: units>
Primary Road	291.741718 [mi]
Secondary Road	175.979652 [mi]
Dirt Road	103.734244 [mi]
Tertiary Road (Residential)	42.502989 [mi]
Campground Road	31.066399 [mi]
NPS Service Road	14.100675 [mi]
Parking Lot or Pullout	6.109527 [mi]
Gravel Road	3.176868 [mi]
Tunnel	1.645088 [mi]

9 rows

Import the Campgrounds and Cell Towers

Hide

```
yose_campgrounds_utm <- st_read("./data", layer="yose_poi") |>
  st_transform(epsg_utm11n_nad83) |>
  filter(POITYPE == 'Campground') |>
  select(POINAME)
```

```
Reading layer `yose_poi' from data source
  `D:\Workshops\R-Spatial\rspatial_mod\outputs\rspatial_bgs23\notebooks\data' using driver `ESRI Shapefile'
Simple feature collection with 2720 features and 30 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 246416.2 ymin: 4153717 xmax: 301510.7 ymax: 4208419
Projected CRS: NAD83 / UTM zone 11N
```

Hide

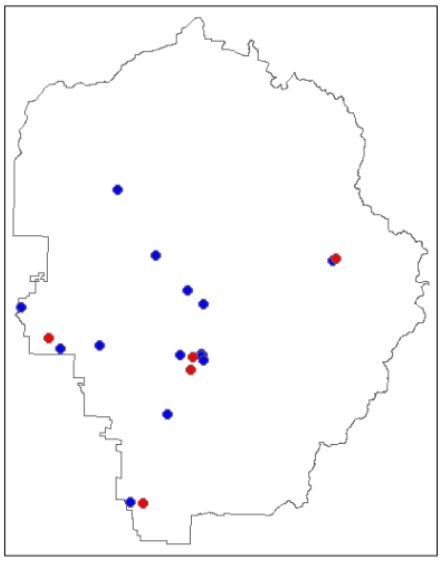
```
yose_celltwrs_utm <- st_read("./data/yose_communications.gdb", "Cell_Towers") |>
  st_transform(epsg_utm11n_nad83)
```

```
Reading layer `Cell_Towers' from data source
  `D:\Workshops\R-Spatial\rspatial_mod\outputs\rspatial_bgs23\notebooks\data\yose_communications.gdb'
  using driver `OpenFileGDB'
Simple feature collection with 5 features and 6 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 251532.4 ymin: 4158756 xmax: 293307.2 ymax: 4194328
Projected CRS: NAD83 / UTM zone 11N
```

Plot them:

Hide

```
tm_shape(yose_bnd_utm) +
  tm_borders() +
tm_shape(yose_campgrounds_utm) +
  tm_symbols(size = 0.3, col = "blue") +
tm_shape(yose_celltwrs_utm) +
  tm_symbols(size = 0.3, col = "red")
```



Compute Distances Between Features - 1 sf data frame

`st_distance()` can be used to compute distances between features. If you pass it one sf object, it will return a symmetric distance matrix between all pairs of features.

Compute the distance between all pairs of campgrounds:

Hide

```
campgrounds_dist_mat <- st_distance(yose_campgrounds_utm)
campgrounds_dist_mat
```

```
Units: [m]
          1        2        3        4        5        6        7        8        9       10        11
1     0.000  8275.796 12628.738 20897.961 24202.415 26366.307 45699.01 26318.172 32505.98 24022.360 27053.5366
2  8275.796     0.000  5634.176 19310.626 20243.499 21669.864 41526.22 18231.678 24574.95 17323.278 20522.7859
3 12628.738  5634.176     0.000 15448.311 15106.794 16243.769 36084.79 14050.939 23230.46 11754.726 14931.6215
4 20897.961 19310.626 15448.311     0.000  6776.104  9775.471 25681.09 23140.503 36074.82 14860.874 15706.6234
5 24202.415 20243.499 15106.794  6776.104     0.000  3014.800 21555.06 18297.281 31962.27  9501.164  9443.6883
6 26366.307 21669.864 16243.769  9775.471  3014.800     0.000 19858.20 16911.114 30774.47  8198.298  7236.6861
7 45699.014 41526.218 36084.793 25681.086 21555.058 19858.200     0.00 32839.145 45873.18 26134.546 23327.6854
8 26318.172 18231.678 14050.939 23140.503 18297.281 16911.114 32839.14     0.000 13892.94  8814.694 10183.5588
9 32505.983 24574.955 23230.456 36074.816 31962.268 30774.468 45873.18 13892.936     0.00 22606.591 24037.3432
10 24022.360 17323.278 11754.726 14860.874  9501.164  8198.298 26134.55  8814.694 22606.59     0.000  3222.2366
11 27053.537 20522.786 14931.621 15706.623  9443.688  7236.686 23327.69 10183.559 24037.34  3222.237     0.0000
12 27174.786 20589.170 15009.085 15972.557  9717.027  7503.431 23439.68  9979.152 23820.38  3268.848   273.4135
13 27088.787 20464.194 14892.172 16067.174  9849.639  7671.851 23645.40  9768.727 23613.03  3140.976   440.3027
14 27616.957 20889.582 15345.375 16750.368 10495.364  8244.331 23684.28  9487.701 23272.44  3604.361  1053.9344
15 22073.083 24539.640 22864.646 11107.445 17786.365 20717.442 32873.41 33489.566 45540.25 25705.918 26791.2639
          12         13         14       15
1  27174.7857 27088.7873 27616.9569 22073.08
2  20589.1697 20464.1936 20889.5823 24539.64
3  15009.0854 14892.1717 15345.3754 22864.65
4  15972.5573 16067.1742 16750.3676 11107.45
5   9717.0274  9849.6393 10495.3635 17786.37
6   7503.4311  7671.8512  8244.3308 20717.44
7  23439.6793 23645.4050 23684.2839 32873.41
8   9979.1521  9768.7266  9487.7011 33489.57
9  23820.3830 23613.0283 23272.4414 45540.25
10  3268.8477  3140.9757  3604.3606 25705.92
11   273.4135   440.3027  1053.9344 26791.26
12     0.0000   212.8079   782.1046 27054.79
13   212.8079     0.0000   692.4943 27142.02
14   782.1046   692.4943     0.0000 27829.09
15 27054.7878 27142.0250 27829.0938     0.00
```

Compute Distances Between Features - 2 sf data frames

If you pass two sf objects to `st_distance()`, it will return a *n x m* matrix containing the distance between all pairs of features.

Compute the distance between all pairs of campgrounds and cell towers:

Hide

```
cgct_dist_mat <- st_distance(yose_campgrounds_utm, yose_celltwrs_utm)
cgct_dist_mat
```

```
Units: [m]
           [,1]      [,2]      [,3]      [,4]      [,5]
 [1,] 46219.4109 25884.856 26225.834 33538.244  5903.577
 [2,] 42063.8195 19210.673 19163.004 25491.865  2384.789
 [3,] 36624.3349 13645.182 13737.613 23773.991  7499.216
 [4,] 26168.3039 15613.133 17308.435 36061.840 19627.095
 [5,] 22082.7907  9689.048 11543.363 31666.279 21313.320
 [6,] 20396.4206  7834.371  9723.708 30321.762 22987.367
 [7,]   541.1383 24682.675 25995.385 44830.662 42783.486
 [8,] 33352.8435  9178.442  7376.002 13416.944 20585.520
 [9,] 46361.8785 23070.914 21252.438  1824.674 26769.154
[10,] 26671.4314  1890.457  2646.959 22220.557 19249.627
[11,] 23859.2190  1411.166  2821.810 23427.597 22413.196
[12,] 23970.1116  1403.475  2639.230 23199.118 22497.225
[13,] 24175.5888  1257.224  2426.427 22995.829 22383.935
[14,] 24211.1160  1733.602  2314.224 22611.632 22843.581
[15,] 33272.4448 26622.758 28245.217 45778.512 23754.044
```

CHALLENGE: Find the closest cell tower

For the first campground (row), which is the closest cell tower? Answer

Hint: use `which.min()`

Hide

```
which.min(cgct_dist_mat[1,])
```

```
[1] 5
```

Hide

```
## Or for all campgrounds:
apply(cgct_dist_mat, 1, which.min)
```

```
 [1] 5 5 5 2 2 2 1 3 4 2 2 2 2 2 5
```

End

Congratulations, you've completed the Notebook!

To view your Notebook at HTML, save it (again), then click the 'Preview' button in the RStudio toolbar.