Import Vector Data

Code

This Notebook will demonstrate how to import various types of vector GIS data into R.

View Layers for Yosemite National Park

First let's look at layers in the data folder, by passing the directory to `st_layers()` from the `sf` package. This will show us the Shapefiles but not layers that are in 'containers', like file geodatabases, geojson files, etc.

Hide

```
library(sf)

## View spatial layers in the data folder.
st_layers("./data")
```

```
Driver: ESRI Shapefile
Available layers:
```

layer_name <chr>	geometry_type <chr>	features <dbl>	fields <dbl>	crs_name <chr>
sf_schools	Point	445	15	WGS 84
veg37	Polygon	6512	26	NAD83 / UTM zone 11N
yose_boundary	Polygon	1	11	North_American_Datum_1983
yose_poi	Point	2720	30	NAD83 / UTM zone 11N
veg37_alliances	NA	39	3	*NA*

5 rows

Import a Shapefile

Import the 'yose_boundary' layer (a Shapefile)

Hide

```
yose_bnd_ll <- st_read(dsn="./data", layer="yose_boundary")
```

```
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
```

Hide

```
# This also works:
# yose_bnd_ll <- st_read(dsn="./data/yose_boundary.shp")
```

Note 1: we don't need to add the .shp extension

Note 2: this code is using convention to name variables *yose_bnd_ll*.

`yose` - all Yosemite layers start with this
`bnd` - tell me this the park boundary
`ll` - lat-long coordinates

CHALLENGE: View the Object class

Write an expression that returns the class (type) of *yose_bnd_ll*. Answer

Hide

```
## View the class of the object we just created
class(yose_bnd_ll)
```

```
[1] "sf"         "data.frame"
```

We see that `yose_bnd_ll` is both a sf object (simple feature data frame) as well as a data.frame. This means we should be able to use the functions designed for either of those objects.

View the properties of *yose_bnd_ll* by simply running it by itself:

Hide

```
yose_bnd_ll
```

```
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
  UNIT_CODE                                                                  GIS_NOTES
1      YOSE Lands - http://landsnet.nps.gov/tractsnet/documents/YOSE/METADATA/yose_metadata.xml
             UNIT_NAME  DATE_EDIT STATE REGION GNIS_ID     UNIT_TYPE CREATED_BY
```

```
1 Yosemite National Park 2016-01-27    CA     PW  255923 National Park      Lands
                                          METADATA PARKNAME                        geometry
1 http://nrdata.nps.gov/programs/Lands/YOSE_METADATA.xml Yosemite POLYGON ((-119.8456 37.8327...
```

CHALLENGE: Which CRS?

What coordinate reference system is *yose_bnd_ll* in? Answer

ANS. Unprojected geographic coordinates (lon-lat) in the NAD83 datum.

View the Attribute Table

The `names()` function returns the column labels of a data frame (in this case the attribute table).

Hide

```
## View column names in the attribute table
names(yose_bnd_ll)
```

```
 [1] "UNIT_CODE"  "GIS_NOTES"  "UNIT_NAME"  "DATE_EDIT"  "STATE"      "REGION"     "GNIS_ID"    "UNIT_TYPE"
 [9] "CREATED_BY" "METADATA"   "PARKNAME"   "geometry"
```

Take note of the last column - `geometry`. That's where the geometry is saved (we'll come back to that later).

View the first few rows of the attribute table with `head()`:

Hide

```
head(yose_bnd_ll)
```

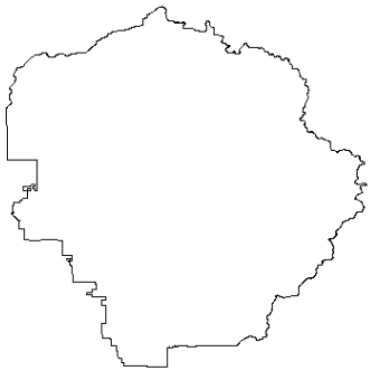
```
Simple feature collection with 1 feature and 11 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -119.8864 ymin: 37.4947 xmax: -119.1964 ymax: 38.18515
Geodetic CRS:  North_American_Datum_1983
  UNIT_CODE                                                                GIS_NOTES
1      YOSE Lands - http://landsnet.nps.gov/tractsnet/documents/YOSE/METADATA/yose_metadata.xml
              UNIT_NAME  DATE_EDIT STATE REGION GNIS_ID     UNIT_TYPE CREATED_BY
1 Yosemite National Park 2016-01-27    CA     PW  255923 National Park      Lands
                                          METADATA PARKNAME                        geometry
1 http://nrdata.nps.gov/programs/Lands/YOSE_METADATA.xml Yosemite POLYGON ((-119.8456 37.8327...
```

Plot the Yosemite Boundary

To plot just the geometry of a sf object (i.e., no symbology from the attribute table), we can use the `st_geometry()` function.

Hide

```
## Plot the geometry (outline) of the Yosemite boundary
plot(yose_bnd_ll |> st_geometry(), asp=1)
```

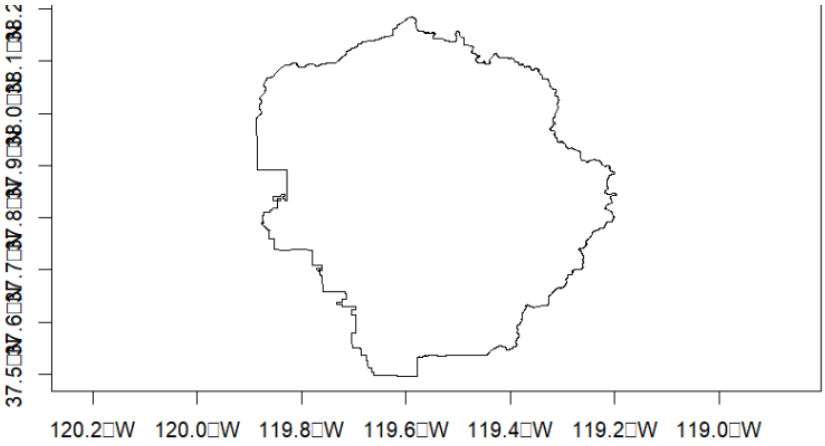


CHALLENGE: Add Axes

Add `axes=TRUE` to your plot() statement. Answer

Hide

```
## Plot the geometry (outline) of the Yosemite boundary
plot(yose_bnd_ll |> st_geometry(), asp=1, axes=TRUE)
```



CHALLENGE: Import POI Shapefile

Import the Yosemite Points-of-Interest (POI) Shapefile and plot them. Answer

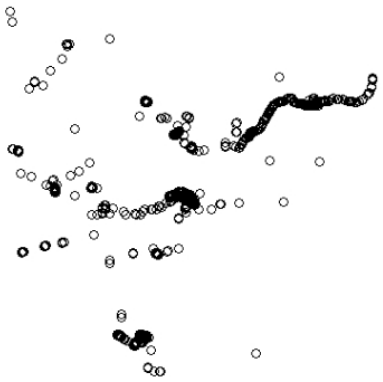
Hide

```
## Import the yose_poi.shp
yose_poi_ll <- st_read(dsn="./data", layer="yose_poi")
```

```
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

```
plot(yose_poi_ll |> st_geometry())
```



Import a KML

kml & kmz files can have more than one layer. Hence the source is the kml file, and you must specify the layer by name.

Import a kml containing the National Register of Historic Places in Yosemite in Yosemite. First find the KML file:

Hide

```
## Import KML file
kml_fn <- "./data/yose_historic_pts.kml"
file.exists(kml_fn)
```

```
[1] TRUE
```

View the layers within this KML:

Hide

```
## View the layers in this kml
```

```
st_layers(kml_fn)
```

```
Driver: KML
Available layers:
```

layer_name	geometry_type	features	fields	crs_name
<chr>	<chr>	<dbl>	<dbl>	<chr>
yose_historic_places	Point	35	2	WGS 84

1 row

Import:

Hide

```
## Import the 'yosem_historic_places' layer
yose_hp_ll <- st_read(kml_fn, layer="yose_historic_places")
```

```
Reading layer `yose_historic_places' from data source
  `D:\Workshops\R-Spatial\rspatial_mod\outputs\rspatial_bgs23\notebooks\data\yose_historic_pts.kml'
  using driver `KML'
Simple feature collection with 35 features and 2 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: -119.8447 ymin: 37.51356 xmax: -119.2165 ymax: 38.08368
Geodetic CRS:  WGS 84
```

View its properties:

Hide

```
## View properties
yose_hp_ll
```

```
Simple feature collection with 35 features and 2 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: -119.8447 ymin: 37.51356 xmax: -119.2165 ymax: 38.08368
Geodetic CRS:  WGS 84
First 10 features:
                              Name Description                  geometry
1  Hetch Hetchy Railroad Engine No. 6             POINT (-119.786 37.67437)
2             Hodgdon Homestead Cabin              POINT (-119.656 37.53924)
3                      Rangers' Club             POINT (-119.5883 37.74735)
4                   Buck Creek Cabin             POINT (-119.4897 37.56131)
5               Wawona Covered Bridge              POINT (-119.656 37.53859)
6             Crane Flat Fire Lookout             POINT (-119.8207 37.75978)
7      Glacier Point Trailside Museum             POINT (-119.5731 37.72916)
8                     McCauley Cabin             POINT (-119.3676 37.87812)
9                 Bagby Stationhouse             POINT (-119.7862 37.67439)
10                 Great Sierra Mine              POINT (-119.2688 37.9276)
```

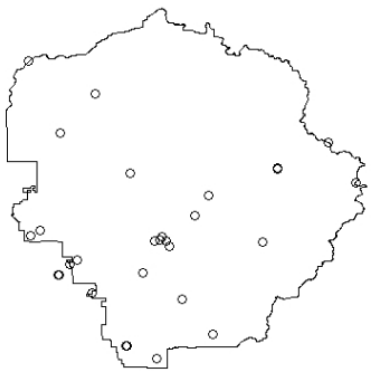
Plot the Historic Places on top of the Park Boundary

Remember to overlay more than one layer on a plot:

- both layers must have the same CRS
- include add=TRUE to the plot statements

Hide

```
## Plot the boundary, then the historic places
{plot(yose_bnd_ll |> st_geometry(), asp=1)
plot(yose_hp_ll |> st_geometry(), add=TRUE)}
```



Import a GeoJSON file

Import a GeoJSON file

Import the California county boundaries, which is saved as a GeoJSON file.

Hide

```
## Import a Geojson file
counties_fn <- "./data/ca_counties.geojson"
file.exists(counties_fn)
```

```
[1] TRUE
```

View the layers in this GeoJSON file:

Hide

```
## View the layers
st_layers(counties_fn)
```

```
Driver: GeoJSON
Available layers:
```

layer_name <chr>	geometry_type <chr>	features <dbl>	fields <dbl>	crs_name <chr>
ca_counties	3D Multi Polygon	58	13	WGS 84

1 row

Import the 'ca_counties' layer:

Hide

```
## Import the 'ca_counties' layer
ca_counties_ll <- st_read(counties_fn)
```

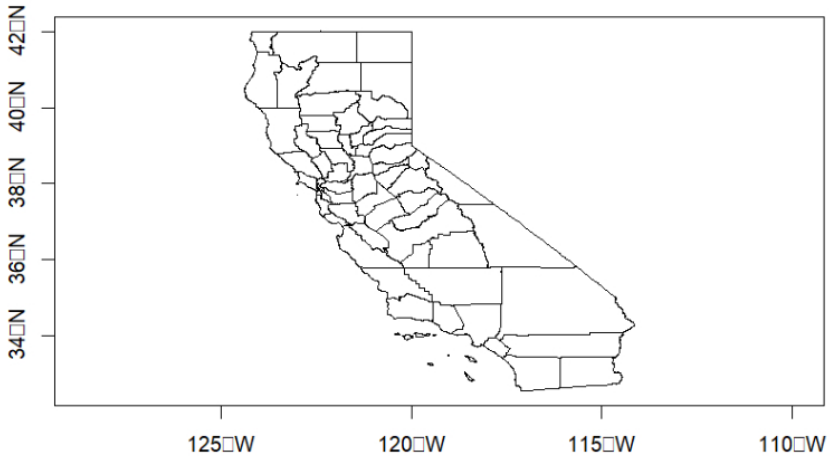
```
Reading layer `ca_counties' from data source
  `D:\Workshops\R-Spatial\rspatial_mod\outputs\rspatial_bgs23\notebooks\data\ca_counties.geojson'
  using driver `GeoJSON'
Simple feature collection with 58 features and 13 fields
Geometry type: MULTIPOLYGON
Dimension:     XYZ
Bounding box:  xmin: -124.4096 ymin: 32.53416 xmax: -114.1312 ymax: 42.00952
z_range:       zmin: 0 zmax: 0
Geodetic CRS:  WGS 84
```

CHALLENGE: Plot Counties

Plot the county boundaries. Answer

Hide

```
plot(ca_counties_ll |> st_geometry(), asp=1, axes=TRUE)
```



Import from a Geodatabase

You can import (but not write to) an ESRI file geodatabase using the sf package. In this case, the source is the folder containing the geodatabase.

Import the Yosemite's trails from a geodatabase. First find the gdb file:

Hide

```
## Define the path to the file geodatabase (a folder)
gdb_fn <- "./data/yose_trails.gdb"
file.exists(gdb_fn)
```

```
[1] TRUE
```

View the layers in this source:

Hide

```
st_layers(gdb_fn)
```

```
Driver: OpenFileGDB
Available layers:
```

layer_name <chr>	geometry_type <chr>	features <dbl>	fields <dbl>	crs_name <chr>
Trailheads	Point	73	5	NAD83 / UTM zone 11N
Trails	Multi Line String	1074	13	NAD83 / UTM zone 11N
Wilderness_stock_routes	Multi Line String	11	6	NAD83 / UTM zone 11N
Winter_trails	Multi Line String	24	5	NAD83 / UTM zone 11N
T_1_DirtyAreas	Multi Polygon	1	3	NAD83 / UTM zone 11N
T_1_PointErrors	Point	260	7	NAD83 / UTM zone 11N
T_1_LineErrors	Multi Line String	0	8	NAD83 / UTM zone 11N
T_1_PolyErrors	Multi Polygon	0	9	NAD83 / UTM zone 11N

8 rows

Import the 'Trails' layer

Hide

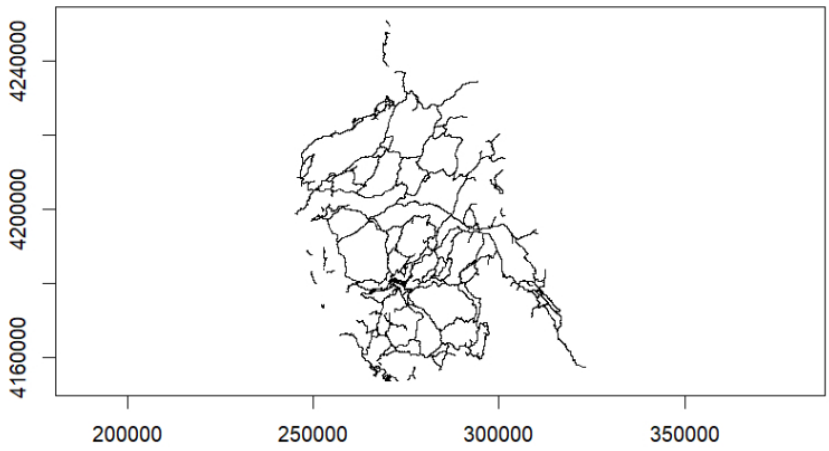
```
## Import the 'Trails' layer  (case sensitive!)
yose_trails <- st_read(gdb_fn, layer="Trails")
```

```
Reading layer `Trails' from data source
  `D:\Workshops\R-Spatial\rspatial_mod\outputs\rspatial_bgs23\notebooks\data\yose_trails.gdb'
  using driver `OpenFileGDB'
Simple feature collection with 1074 features and 13 fields
Geometry type: MULTILINESTRING
Dimension:     XY
Bounding box:  xmin: 245134 ymin: 4153668 xmax: 323239.7 ymax: 4250703
Projected CRS: NAD83 / UTM zone 11N
```

Plot Yosemite's Trails:

Hide

```
## Plot the trails layer
plot(st_geometry(yose_trails), axes=TRUE)
```



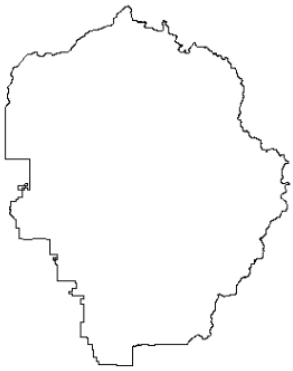
CHALLENGE: Diagnose a Bad Plot

The following code does **not** work to make a plot of the park boundary and the trails. Can you tell why? Answer

ANS. Because the CRS are different!

Hide

```
{plot(yose_bnd_ll |> st_geometry())
plot(yose_trails |> st_geometry(), add=TRUE)}
```



Import from a GeoPackage

Let's import Yosemite's watersheds from a geopackage file.

Hide

```
## Import watersheds from a geopackage
gpkg_watershd_fn <- "./data/yose_watersheds.gpkg"
file.exists(gpkg_watershd_fn)
```

```
[1] TRUE
```

Hide

```
st_layers(gpkg_watershd_fn)
```

```
Driver: GPKG
Available layers:
```

layer_name <chr>	geometry_type <chr>	features <dbl>	fields <dbl>	crs_name <chr>
calw221	Polygon	127	12	NAD83 / California Albers

1 row

Hide

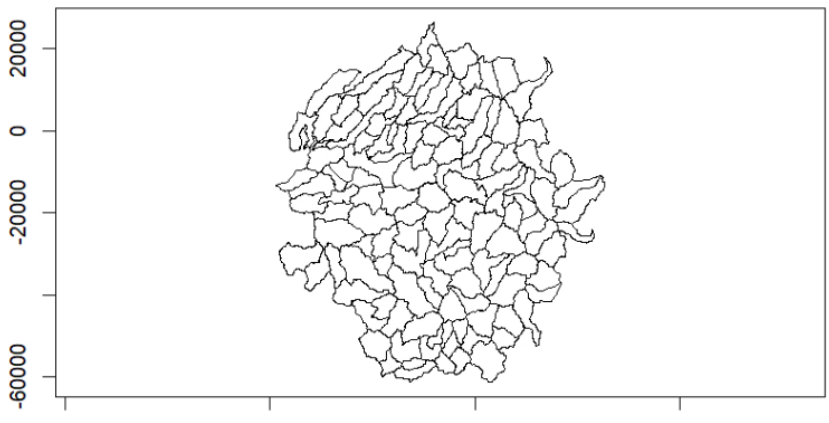
```
yose_watersheds <- st_read(gpkg_watershd_fn, layer="calw221")
```

```
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
```

Plot the watersheds:

Hide

```
plot(st_geometry(yose_watersheds), axes=TRUE)
```



-50000 0 50000 100000

CHALLENGE: What CRS?

What CRS are the Yosemite watersheds in? Answer

ANS. California Equal Albers (a common projection for statewide data in California)

Hide

```
st_crs(yose_watersheds)
```

```
Coordinate Reference System:
  User input: NAD83 / California Albers
  wkt:
PROJCRS["NAD83 / California Albers",
    BASEGEOGCRS["NAD83",
        DATUM["North American Datum 1983",
            ELLIPSOID["GRS 1980",6378137,298.257222101,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4269]],
    CONVERSION["California Albers",
        METHOD["Albers Equal Area",
            ID["EPSG",9822]],
        PARAMETER["Latitude of false origin",0,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8821]],
        PARAMETER["Longitude of false origin",-120,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8822]],
        PARAMETER["Latitude of 1st standard parallel",34,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8823]],
        PARAMETER["Latitude of 2nd standard parallel",40.5,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8824]],
        PARAMETER["Easting at false origin",0,
            LENGTHUNIT["metre",1],
            ID["EPSG",8826]],
        PARAMETER["Northing at false origin",-4000000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8827]]],
    CS[Cartesian,2],
        AXIS["easting (X)",east,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["northing (Y)",north,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    USAGE[
        SCOPE["State-wide spatial data management."],
        AREA["United States (USA) - California."],
        BBOX[32.53,-124.45,42.01,-114.12]],
    ID["EPSG",3310]]
```

CHALLENGE: Import another Layer

Look at the other GIS files in the data folder. Select one, import it, and plot it.

Hide

```
## Your answer here
```