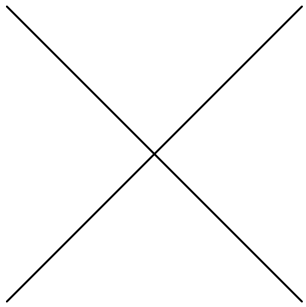
Gerrymandering

Practice related to this lecture is found in the worksheet associated with Project: Gerrymandering.

Spatial II

The following exercises serve as an introduction to the functionality of the `sf` package. Anytime you use a new function, use `?` to learn how it works. Some of these exercises ask you to plot the result. Plots are provided to check your answer.

1. Following the approach used in lecture, use `st_linestring` to create two separate line segments called `l1` that stretches from $(0, 0)$ to $(1, 1)$ and `l2` that stretches from $(0, 1)$ to $(1, 0)$. View them using base R `plot()`[1].



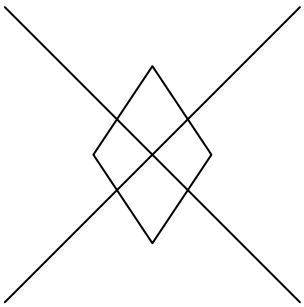
2. Form a spatial data frame (an object with classes `sf` and `data.frame`) called `my_lines` with two columns:

1. `id`: an attribute with values `1` and `2`.
2. `geom`: a *simple feature (geometry) column* containing the two lines. with the coordinate reference system OGC:CRS84.

Use the coordinate reference system "OGC:CRS84".

[1]Each object will require a separate call to `plot()`. You can add `add = TRUE` to `plot()` so that the contents of the second get added to the first. Note that these plot additions don't carry over from code cell to code cell.

3. Create a second spatial data frame, this one with a diamond-shaped polygon. The diamond should have corners at $(.5, .8)$, $(.3, .5)$, $(.5, .2)$, and $(.7, .5)$ and use the same crs. It should have an attribute called `price` with the value `100`. Call your spatial data frame `my_diamond` and plot it atop your lines.



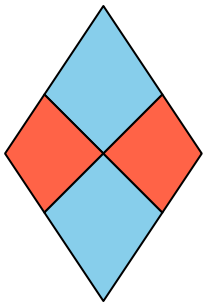
4. Calculate the dimension, length, and area of `my_lines()`.

5. Calculate the dimension, length, and area of `my_diamond()`.

6. Use the `st_split()` function inside the `lwgeom` package to split the diamond into four diamonds using your lines and name the resulting object `split_diamond`. After inspecting the result, run the following line to extract each of the polygons into a separate feature (row) of a spatial data frame.

```
split_diamond <- split_diamond |>
    st_collection_extract("POLYGON")
```

Plot the resulting sf collection of 4 features, with each of the split diamonds filled with a color[2].



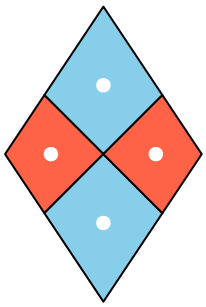
7. Add two attribute columns to `split_diamonds`:

1. `yield`: that takes values of 10, 20, 30, and 40 for the top, right, bottom, and left split diamonds, respectively. You can imagine this is the yield of four fields of corn.
2. `split_price`: the original `price` of the full diamond scaled by the relative area of each of the four split diamonds. Round the split price to the nearest cent.

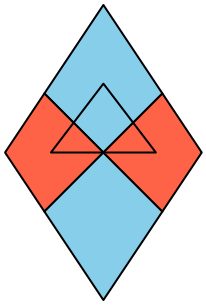
Be sure to save the new columns back into `split_diamonds`.

[2]You can add colors to `plot()` using the `col` argument and call `colors()` at the console to see the color names available in base R. I used `col = c("skyblue", "tomato")` but you're welcome to pick your own colors.

8. Compute the centroids of the four split diamonds, call them `my_cents`, and add them to your plot. Set their `col` and `pch` as you like.

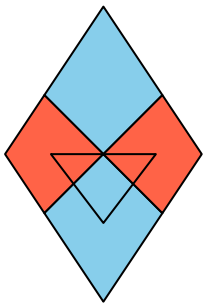


9. Using what you know about data frame subsetting, remove the bottom-most centroid. Then take the union of the three remaining points, form a convex hull from them, and save the polygon to `my_tri_up`[3]. Plot `my_tri_up` on top of your split diamond.



[3]The functions for these set operations in `sf` are exactly as you'd expect: the name of the operation prepended by `st_`. Note also that the union operation will drop the `sf` class. You'll need to run your object through `st_sf()` to get it back.

10. Repeat the previous exercise, but form an sf object called `my_tri_down` that has the top-most centroid removed.



11. Under the definition of "intersects", which of your split diamonds intersect the `my_tri_up`? Check by doing a spatial join.

12. Calculate the area of `my_tri_up` two ways:

1. By directly measuring the area of the triangle.
2. By interpolating from the `area` attribute of the split diamonds that it intersects with[4].

Do they agree?

13. Calculate the `yield` of both `my_tri_up` and `my_tri_down` by interpolating from the `yield` of the split diamonds that they intersect with. Why are they different from one another?

[4]When interpolating, you'll need to decide whether your variable is *spatially extensive*. An attribute that is spatially extensive has a magnitude that is proportional to the size of the feature it's measured on; as the area of the feature increases, the magnitude of the attribute increase propotionally.

Dashboards

Starting from the contents of `demo-dashboard.qmd` found in the corresponding repo (https://github.com/berkeley-stat133/demo-dashboard) create a data dashboard the provides information on the most recent earthquakes to strike New Zealand. The dashboard should include:

1. Your name added as the author.
2. A theme of your choosing[5].
3. A layout in two columns.
4. The map of New Zealand taking up the full right column.
5. The left of column should two rows, the top of which has value boxes for the number of hours since the last earthquake and the number of earthquakes in the last 24 hours. Select a suitable icon for each[6].
6. The bottom row should hold the table of the top n earthquakes.
7. All of the above should be on a page called "Dashboard". Add a second page called "About the data" that displays the following:

   The original author of this dashboard is Charlotte Wickham.

   The data on earthquakes is drawn from the GeoNet API:

   https://api.geonet.org.nz/

   All earthquakes below MMI 3.0 have been filtered out.

Since it is quite lengthy, there is no need to copy the code used in your dashboard into this problem set.

[5]Dashboards collect the colors and fonts into themes similar to ggplot2. You can find the list of themes here: https://quarto.org/docs/dashboards/theming.html.

[6]Quarto dashboards can use any of the icons in a large collection called Bootstrap. Search for an icon and then use the corresponding name as a string: https://icons.getbootstrap.com/.