Map of Surviving Historic Landmarks in Orleans Parish

Procedure Overview In this section, the goal is to build an interactive map of Orleans Parish, showing historic and geographic features to help us better understand the area.

To do so, we'll use GeoJSON data showing historic landmarks and jazz musician homes, downloaded from a New Orleans government website. The jazz homes data will be loaded without transformation into a map layer using kepler.gl, but the landmarks data will need to undergo a process of flattening and processing, in order to extract dates from it. With properly formatted dates, the data can then be used in the kepler.gl application to add a dimension of time to the map.
This procedure will also allow us to do some basic analysis of the data, and to draw some interesting conclusions about Orleans Parish and its history.

Procedure Detail Use the geojsonR and jsonlite packages to load the geoJSON files.

```
source <- "https://data.nola.gov/Geographic-Base-Layers/Local-Landmarks/srrj-xwma"
f <- 'untidyData/Local Landmarks.geojson'
landmarks <- FROM_GeoJson(f)
class(landmarks)
```

```
## [1] "list"
```

```
names(landmarks)
```

```
## [1] "features" "type"
```

What do these hold?

```
c(length(landmarks$features), length(landmarks$type))
```

```
## [1] 1242    1
```

With JSON objects, or any objects, for that matter, it's often best to inspect 1242 things cautiously:

```
str(landmarks$features[[1]])
```

```
## List of 3
##  $ geometry  :List of 2
##   ..$ type       : chr "Point"
##   ..$ coordinates: num [1:2] -90.1 29.9
##  $ properties:List of 14
##   ..$ addl_addr : chr ""
##   ..$ architect : chr "James Freret"
##   ..$ const_date: chr "1868"
##   ..$ geo_addr  : chr "1641 Amelia Street"
```

```
##   ..$ name      : chr "Hernandez-Davis House"
##   ..$ no_cbd    : chr "New Orleans"
##   ..$ nom_des   : chr "Designated"
##   ..$ num1_edit : chr "1641.0"
##   ..$ num2_edit : chr "0.0"
##   ..$ num_orig  : chr "1641"
##   ..$ objectid  : chr "1"
##   ..$ staff     : chr "BDB"
##   ..$ str_orig  : chr "Amelia Street"
##   ..$ street    : chr "Amelia Street"
##  $ type      : chr "Feature"
```

So each item in the feature list (1242 of them) is one landmark. Each of these comprises a `type` field ("Feature", self-referentially), another nested structure `geometry`, describing its own `type` as `Point` or `MultiPolygon` for some items, and listing its two `coordinates`, and then a bunch of `properties` like architect, address, year of construction, etc.

Build a utility function in hopes of saving some time further down the line when processing other GeoJSON files.

```
# this function takes a geojson file as input and returns a list of 2 data.frames,
## one for the Points, and one for the MultiPolygons
geojson2table <- function(gjfile) {
  geo <- FROM_GeoJson(gjfile)
  pointframe <- 0  # accumulate Points here
  polyframe  <- 0  # and MultiPolygons here
  for (feat in geo$features){
    if (feat$geometry$type == 'Point') {
      if (!is.data.frame(pointframe)) {
        pointframe <- data.frame(feat)
      } else {
          pointframe <- rbind(pointframe, data.frame(feat))
      }
    } else if (feat$geometry$type == 'MultiPolygon') {  # store these in case
        if (!is.data.frame(polyframe)) {
         polyframe <- data.frame(feat)
        } else {
          polyframe <- rbind(polyframe, data.frame(feat))
        }
      }
  }
  # Transforming the nested structure using data.frame() makes two rows for each
  # observation: 1 for the latitude and one for longitude.  Pivoting these
  # wider doesn't work though, since each lat/lon is an actual value.

  # Plan B: Combine every 2 rows into lat/lon pairs using lead(), and then
  # remove the redundant rows.
  pointframe$lat <- lead(pointframe$geometry.coordinates)
  pointframe$lon <- pointframe$geometry.coordinates
  # now drop every second row, which is a dupe
  pointframe <- pointframe %>%
    filter(row_number() %% 2 == 1)
  list(points = pointframe, polys = polyframe)
}
```

Make 2 frames for Landmarks, using the above function

```
landmarklist <- geojson2table('untidyData/Local Landmarks.geojson')
names(landmarklist)
```

```
## [1] "points" "polys"
```

And write them to csv

```
write.csv(landmarklist$points, 'untidyData/landmarkPoints.csv')
write.csv(landmarklist$polys, 'untidyData/landmarkPolygons.csv')
```

```
landpoints <- read.csv('untidyData/landmarkPoints.csv')
head(landpoints, n = 2)
```

Map construction years to datetimes so that they can be read by kepler.gl

```
##   X geometry.type geometry.coordinates  properties.addl_addr
## 1 1         Point            -90.09479
## 2 2         Point            -90.08886 802 Delachaise Street
##   properties.architect properties.const_date      properties.geo_addr
## 1         James Freret                  1868       1641 Amelia Street
## 2              Unknown                       3445 Annunciation Street
##           properties.name properties.no_cbd properties.nom_des
## 1   Hernandez-Davis House       New Orleans         Designated
## 2 Mystical's boyhood home       New Orleans          Nominated
##   properties.num1_edit properties.num2_edit properties.num_orig
## 1                 1641                    0                1641
## 2                 3445                    0                3445
##   properties.objectid properties.staff properties.str_orig   properties.street
## 1                   1              BDB       Amelia Street       Amelia Street
## 2                   2              BDB Annunciation Street Annunciation Street
##      type      lat       lon
## 1 Feature 29.92820 -90.09479
## 2 Feature 29.91984 -90.08886
```

```
# The first 3 columns aren't going to be needed
landpoints <- landpoints %>%
  select(c(4:ncol(landpoints)))
```

```
# We need datetimes for kepler.gl
#install.packages('datetime')
library(datetime)
library(purrr)
# We need to deal with "c.1850", "1879-1881", etc., and add times
dates <- str_match(landpoints$properties.const_date, '[0-9]{4}')
# helper func
datify <- function(year) {
  y <- year
  if (!is.na(y)) {
    y <- paste(y, '/01/01 01:01', sep='')
    y <- as.datetime(y, format = '%Y/%m/%d %H:%M')
  }
  y
}
dates <- as.datetime(as.numeric(map(dates, datify)))
# Add processed dates to the d.f
landpoints$datetime <- dates
write.csv(landpoints, 'untidyData/landmarkPoints.csv')
# kepler.gl won't allow NA's in date column, so need to filter those out.
dated <- landpoints %>%
  filter(!is.na(landpoints$datetime))

write.csv(dated, 'untidyData/datedLandmarks.csv')
```

```
geo <- read_json('untidyData/Jazz Houses.geojson')
str(geo$features[[1]])
```

```
## List of 3
##  $ type      : chr "Feature"
##  $ properties:List of 4
##   ..$ address          : chr "811 N LIBERTY ST"
##   ..$ musicianfirstname: chr "James"
##   ..$ musiciansurname  : chr "Brown"
##   ..$ objectid         : chr "355"
##  $ geometry  : NULL
```

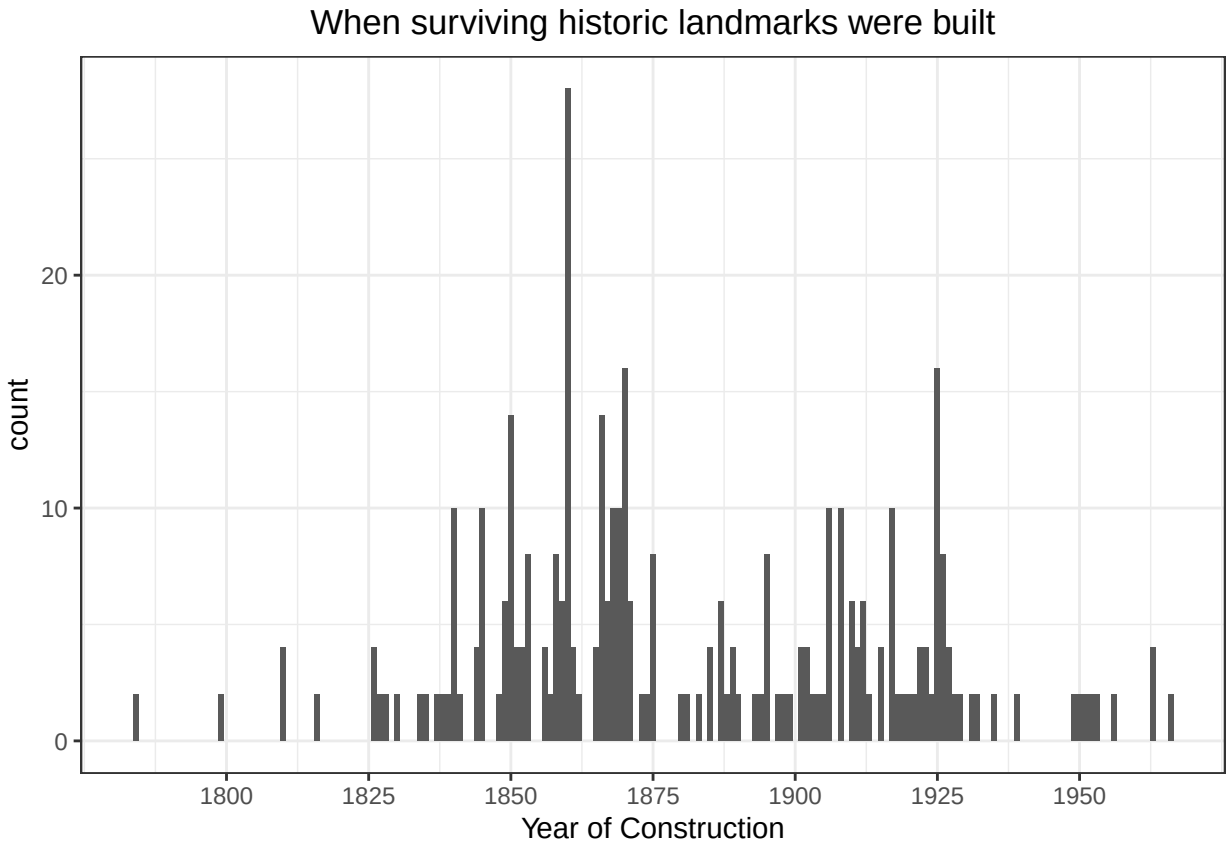
```
## Future todo -- link musicians' birthdays to time-lapse on kepler map.  (Not easy)
```

The rest of the work in building the map takes place over on kepler.gl, and consists of adding layers to the map. The main part for our purposes is to add a filter layer so that the map only shows landmarks constructed within the filtered date range, which can then be part of a time-loop showing the viewer when different areas of the parish were being developed over the last 225 years.

Map with landmarks, jazz houses, Mississippi river, Orleans water features, and parish boundary

Here's a quick look at the distribution of construction dates for surviving landmarks, which is shown at the bottom of the kepler.gl map in the moving timeline.

```
# focus on known dates (about 2/3 of the landmarks)
# Some other day, use KNN (based on lat/lon) to estimate dates for the other 1/3
known_dates <- landpoints %>%
  filter(!is.na(datetime))
# Need to convert long datetimes back to just integer years, to plot
intYears <- as.integer(format(known_dates$datetime, format = '%Y'))
ggplot(data = NULL, aes(intYears)) +
  geom_bar() +
  xlab("Year of Construction") +
  scale_x_continuous(breaks = seq(1800, 1950, by = 25)) +
  ggtitle('When surviving historic landmarks were built') +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```



Since these are historic landmarks, they should tend to be mostly older (which they are), but the 19th century seems to have reached its peak pre-Civil War.
The fact that development drops off sharply after that war suggests an outflow of population from New Orleans when slavery ended.

The next, smaller wave of development looks like it crested in the middle of the 1920's, when New Orleans Jazz was at its peak of popularity, and during an economic boom.

Tools and sources used:

- https://mygeodata.cloud, to convert shapefiles (.shp, amongst other suffixes) to geoJSON MultiPolygons. Those polygons are for mapping anything more complicated than a geo-point (lat/lon), and don't translate well to csv's. Mygeodata is a really useful tool, although they are only free for 3 conversions each month.
- https://Data.NOLA.gov, linked to earlier, has a lot of valuable data, free. It also has a REST API, if you need to make requests for an app, e.g.
- https://kepler.gl is a great way to visualize geo-data.