National Parks Visitation

Charles Coonce

2023-11-13

Getting Familiar with our Data!

We first need to load the data set I found from data.world.

```
  ## Getting my Data Set
data <- read.csv("~/Library/CloudStorage/GoogleDrive-ccoonce@asu.edu/My Drive/DAT 301/ProjectModule4/al
```

First, I would like to understand the structure of this data set.

```
  ## Exploring my Data Set
head(data, 3)
summary(data)
```

This set has 18 columns consisting of 17 'character' variables and one 'int' variable. The various columns contain information like park name, region, state, type of park, and number of visitors. Each observation (or row) is a different park by year.

The Problem

After seeing the structure of this set, I think the first question I would like to answer is which parks I should plan to visit based solely on popularity by number of visitors.

Exploratory Analysis

First, Let's clean some things up by removing unnecessary columns for my analysis.

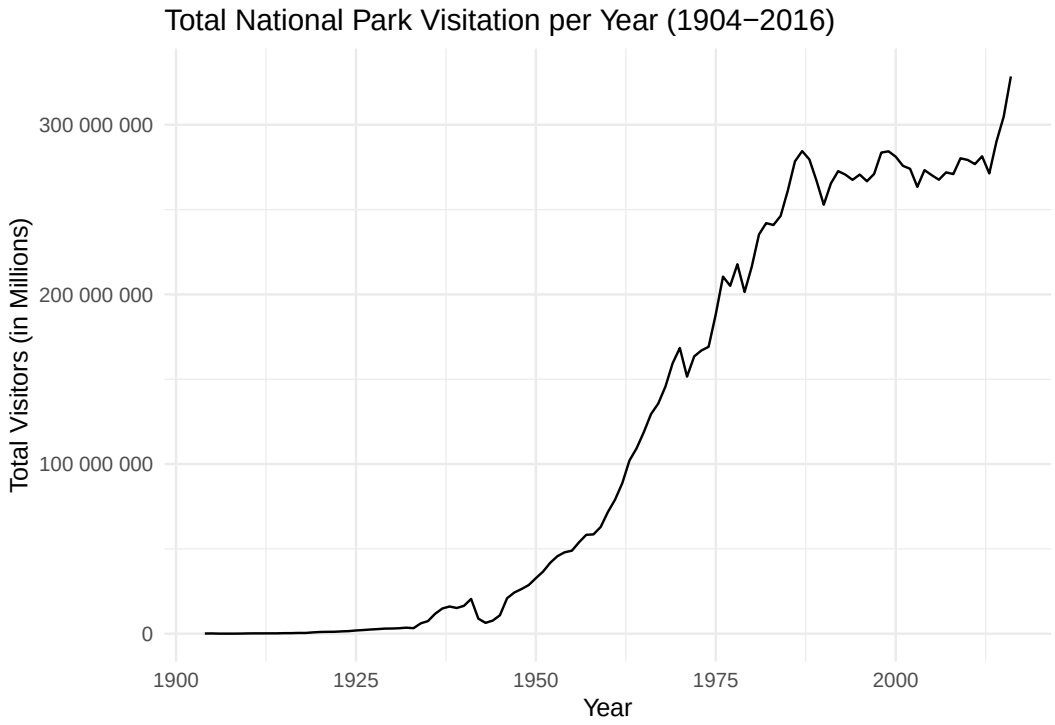
```
  ## Select fewer columns
data_clean <- select(data, region:yearraw)

 ## Check for missing or problem values
year_list <-data_clean %>% count(data_clean$yearraw)
## head(year_list,4)
## tail(year_list,2)
## sum(is.na(data_clean$visitors))
```

After looking at the smaller subset of data, the 'yearraw' column has a 'Total' section that will cause issues during analysis. I will also convert it to a numeric value rather than character value. In addition, the 'visitors' column has 4 values that are not available. I will remove those observations to streamline my exploration.

```r
  ## Removing observations with NA's
data_clean <- na.omit(data_clean)


  ## Subset the data
data_totals <- subset(data_clean, yearraw == "Total")
data_clean <- subset(data_clean, yearraw != "Total")
data_clean$yearraw <- as.numeric(data_clean$yearraw)
```

Visitation Trends

I want to understand the trend for all National park visitation over the years before sub-setting to smaller groups. This will help me compare national trends to the trends of different areas.



This plot shows how visitor numbers have changed over time. It appears there is an overall increasing trend in the number of visitors, which could be attributed to a variety of factors that would require a deeper dive.

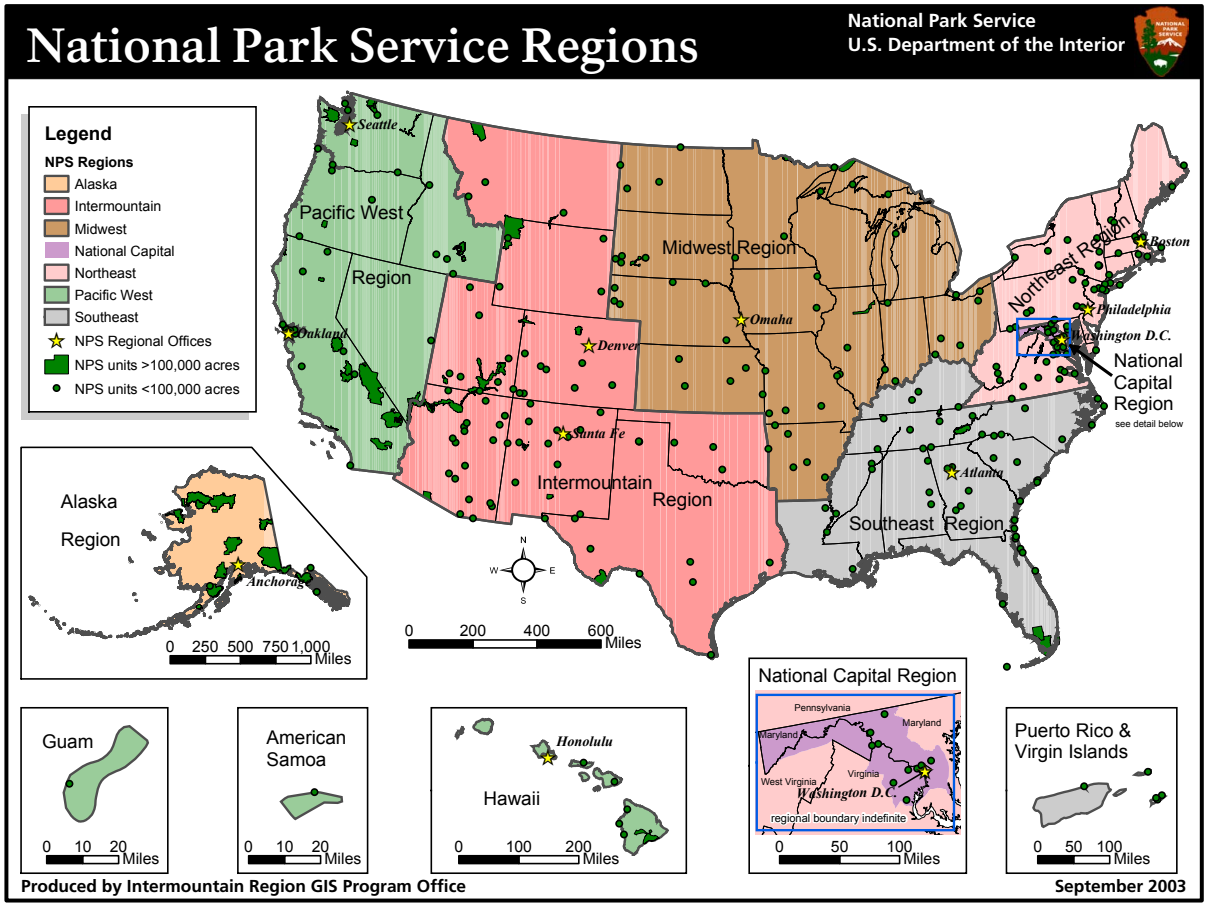


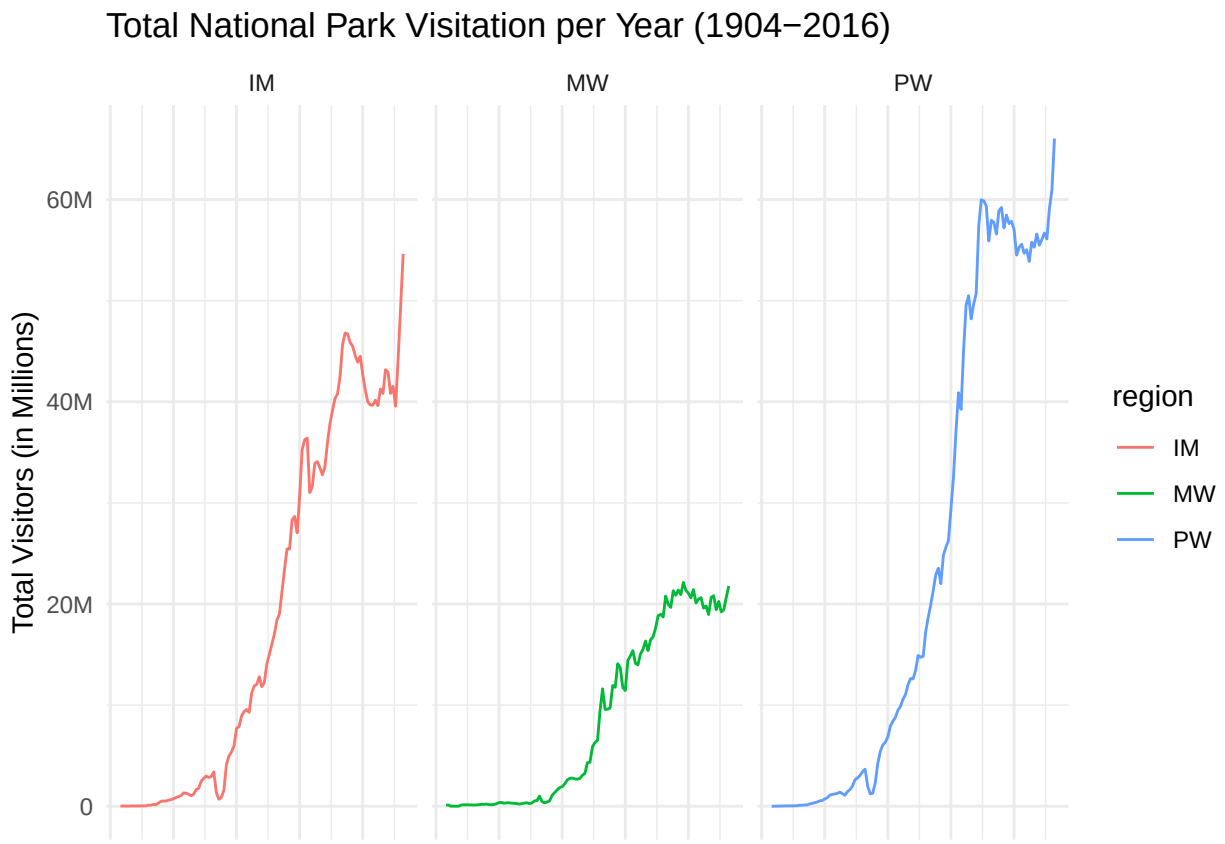
Figure 1: National Parks Service Regions - This poster explains the region categories used in this data set.

Regions

The data set is subdivided into regions as shown in the info graphic above.

I am only interested in regions near me. So I am going to look at the region I am in as well as surrounding regions. I need to group and filter my data set to get the appropriate observations.

```
  ## group and filter by region
region_total <- data_clean %>%
  filter(region == "PW" | region == "IM" |
         region == "MW") %>%
  group_by(yearraw, region) %>%
  summarise(Total = sum(visitors), .groups = "rowwise") %>%
  na.omit()
```



These plots show that Intermountain and Pacific West regions have seen the steepest increases in visits. I will focus my attention on these two regions.

States

In order to show total visitation by state I need to filter and group my data set.

```
  ## Group by State
state_data <- data_clean %>%
  group_by(state) %>%
  summarise(Total = sum(visitors)) %>%
  filter(state %in% c("WA", "OR", "CA", "MT", "WY",
                      "CO", "UT", "NV", "ND", "SD",
                      "NE", "ID", "AZ", "NM"))
```

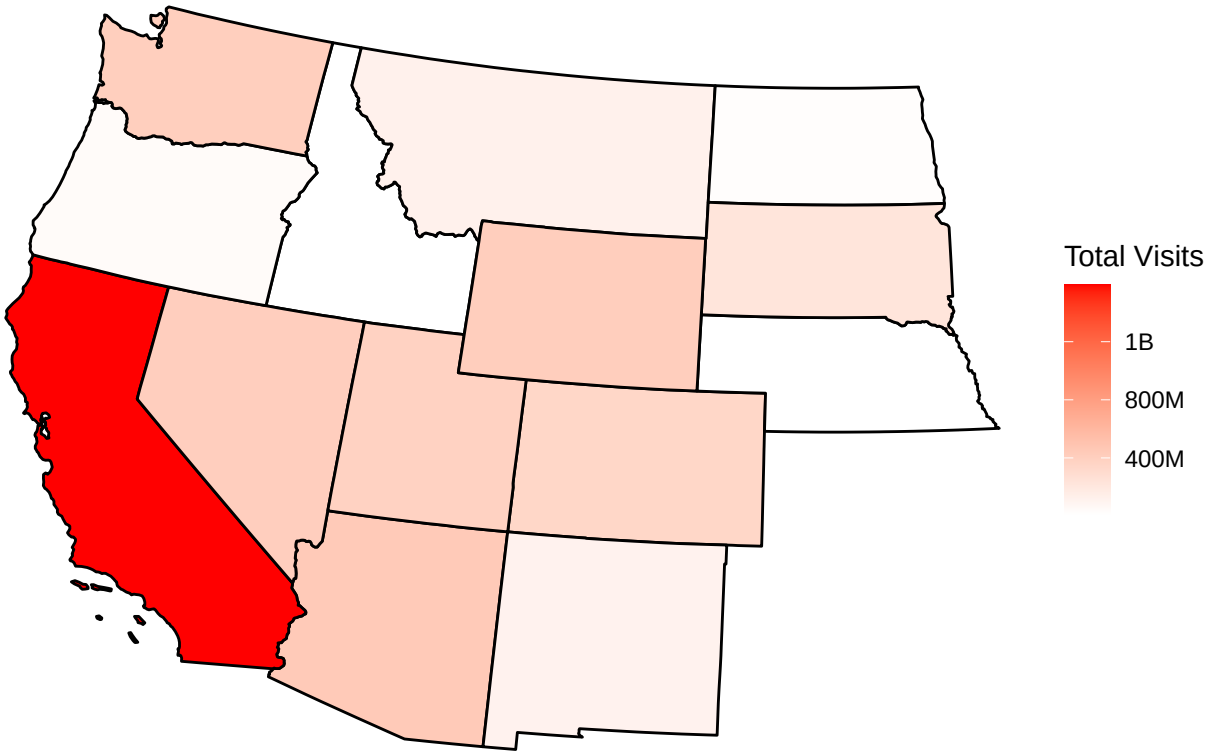
Next, I need to add location data in order to create a heat map of my new data. I can use the 'usmap' package to create an interesting plot.

```
  ## Get the states map data from usmap
states_map <- usmap::us_map(regions = "states") %>%
  filter(abbr %in% c("WA", "OR", "CA", "MT", "WY", "CO",
                     "UT", "NV", "ND", "SD", "NE", "ID",
                     "AZ", "NM"))

  ## Merge visitor data with map data
merged_data <- merge(states_map, state_data, by.x = "abbr",
                     by.y = "state", all.x = TRUE)
```

Now I can prepare my base layer for adding multiple data visualizations.

```
  ## Plot the outlines of the states with heatmap
base_map <- ggplot() +
  geom_polygon(data = merged_data, aes(x = x, y = y, group = group),
               fill = "white", color = "black", linewidth = 0.25)
```

Heatmap of Total National Park Visits by Selected Western US States



This heatmap shows which states have the most visitors. Based on the results, I will pick the top parks in California, Wyoming, and Washington. I'd like to start thinking about weighing the distance from my home and the number of visitors to see if it can help me decide.

I will need to find another dataset to add latitude and longitude values for the top parks. and add that information into my top parks data I started preparing below.

```
top_parks <- data_clean %>%
  group_by(unit_name, state) %>%
  summarise(Total = sum(visitors), .groups = "rowwise") %>%
  arrange(desc(Total)) %>%
  filter(state %in% c("WA", "OR", "CA", "MT", "WY",
                      "CO", "UT", "NV", "ND", "SD",
                      "NE", "ID", "AZ", "NM"))
```

I was able to compile the coordinates for a number of the most visited parks. I need to create a data frame before I can merge my top parks data with these locations.

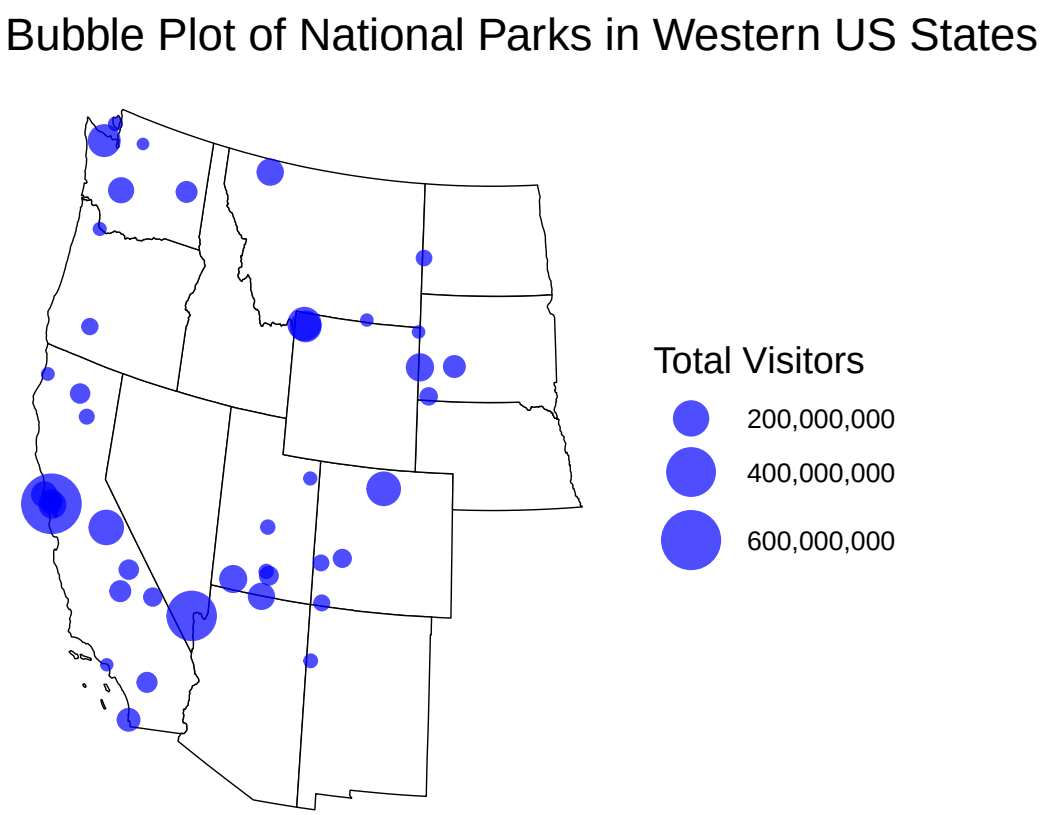
```
  ## Creating a new data frame from the coordinates I compiled
geo_data <- read_csv("geo_data.txt", col_types = cols(Latitude = col_number(),
    Longitude = col_number()))
```

```
 ## adding coordinates to my list of top parks
top_parks <- merge(top_parks, geo_data,
                   by.x = "unit_name", by.y = "Park Name")

  ## rearranging columns to make manipulation easier
top_parks <- select(top_parks, unit_name, state, Total, Longitude, Latitude)
```

After Merging, I now need to convert the Longitude and Latitude to the 'usmap' coordinate system in order to map locations onto my base layer.

```
  ## 'usmap' function to transform Latitude and Longitude to 'usmap' coordinates.
transformed_top_parks <- usmap_transform(top_parks, input_names = c("Longitude", "Latitude"))
```

Now I can create a plot to summarize my findings.



Now that I have a better idea of where specific top parks are I can start narrowing my search. I think I should consider distance from my home.

Distance Analysis

I will start by loading and converting my hometown coordinates to the 'usmap' coordinate system.

```
  ## Coordinates for Helena, Montana
helena_coords <- data.frame(Longitude = -112.0372, Latitude = 46.5891)
helena_coords_transformed <- usmap_transform(helena_coords, input_names = c("Longitude", "Latitude"))
```

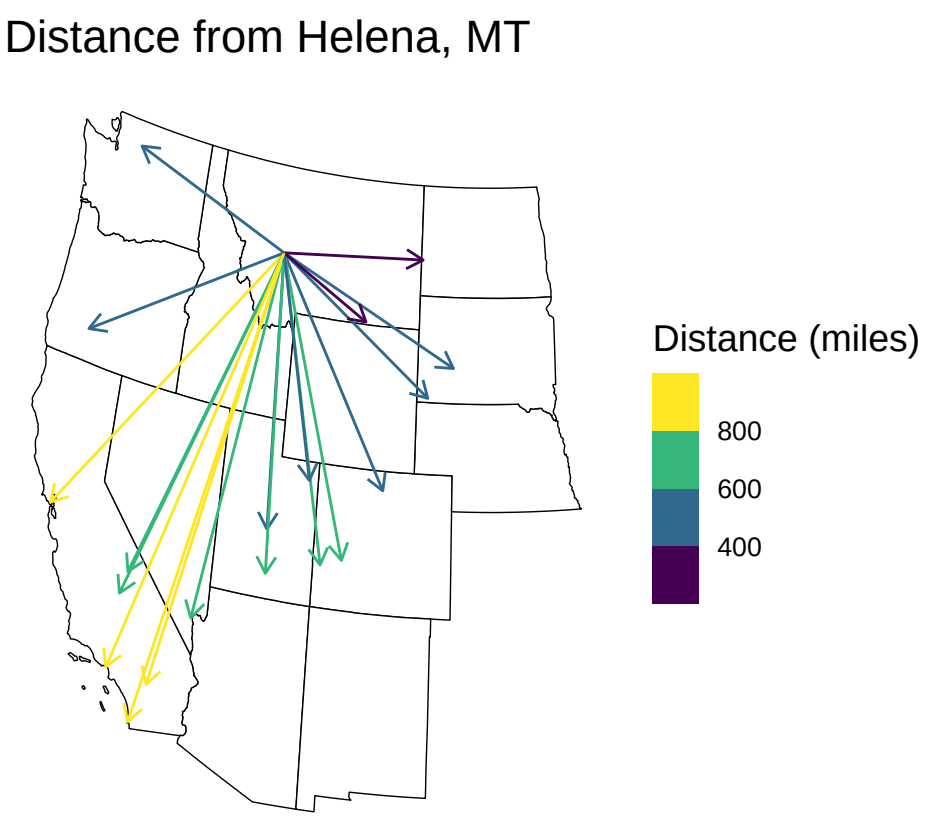
Now I need to use the Haversine formula which is useful for determining the shortest path between two points on the Earth using the Latitude and Longitude coordinate system.

```
  ## Calculate distances from Helena to each park
transformed_top_parks$distance <- distHaversine(matrix(c(helena_coords$Longitude,
                                                         helena_coords$Latitude), nrow = 1),
                                                matrix(c(transformed_top_parks$Longitude,
                                                transformed_top_parks$Latitude),
                                                ncol = 2), r=6378137)
  ## Converting to miles from Km
transformed_top_parks$distance_miles <- transformed_top_parks$distance / 1609.34
```

In order to plot this information I want to add my hometown coordinates to the top parks data frame.

```
 ## that repeats Helena's coordinates
helena_repeated <- data.frame(
  x = rep(helena_coords_transformed$x, nrow(transformed_top_parks)),
  y = rep(helena_coords_transformed$y, nrow(transformed_top_parks))
)

 ## Add Helena's coordinates to the 'transformed_top_parks' data frame
transformed_top_parks$helena_x <- helena_repeated$x
transformed_top_parks$helena_y <- helena_repeated$y
```

And finally, I will create the plot.

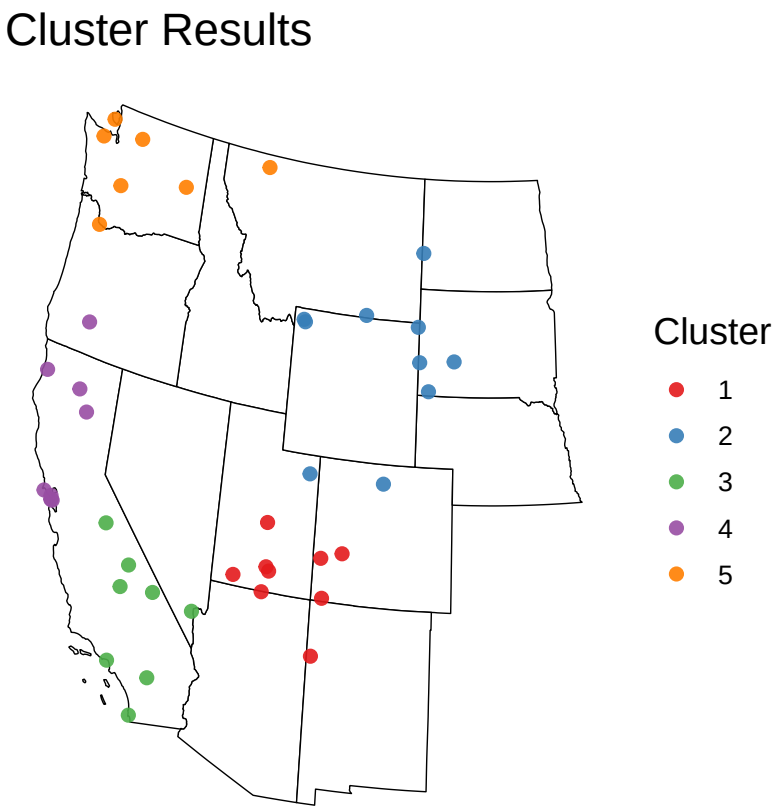


Re-Defining the problem

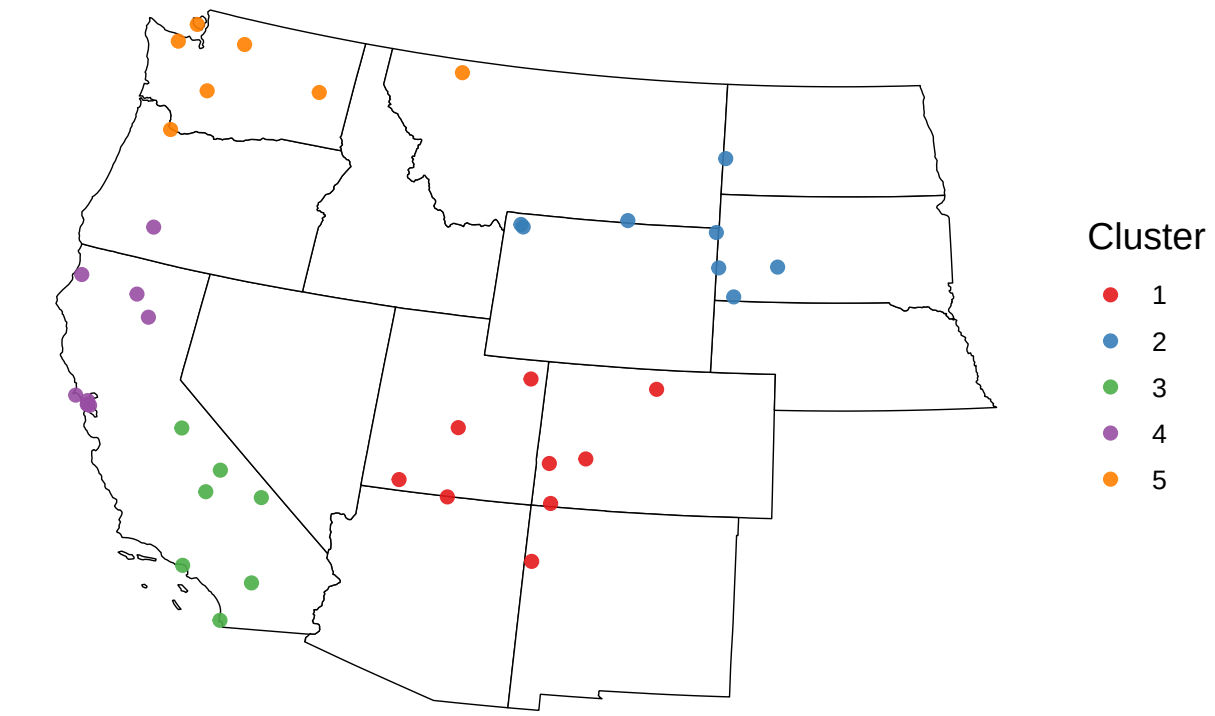
After seeing this visualization, I want to pivot to a new problem. How I can optimize seeing multiple parks per trip rather than trying to choose one park. There are too many nearby! I need to use a clustering model to group the parks by looking at their distances from each other. To do this I need to use a hierarchical cluster function.

```
  ## Perform a hierarchical clustering

  ## Creating distance matrix
dist_matrix <- dist(top_parks_cluster[,c('Longitude', 'Latitude')])
clusters <- hclust(dist_matrix)

  ## Split into clusters of parks
park_clusters <- cutree(clusters, k = 5)
top_parks_cluster$cluster <- park_clusters
```

With my new set of clustered parks, I want to see how the clusters worked out.



Not a bad result, but there are a few parks I want to change to a different cluster.



Better! Now I want to know what the most efficient way to drive from park to park is so I will use the Traveling Salesman Problem (TSP). The TSP finds shortest possible route that visits each city exactly once and returns to the original city. I would likely be able to get a more accurate route optimization if I was using road directions, rather than straight distances. I think that a basic TSP is more than enough for this situation.

```
  ## Initialize an empty list to store results
tsp_results <- list()

  ## Loop through each cluster
for (i in 1:max(top_parks_cluster$cluster)) {
    ## Subset parks for the current cluster
  cluster_parks <- top_parks_cluster[top_parks_cluster$cluster == i, ]

    ## Calculate the distance matrix for the current cluster
  dist_matrix <- as.dist(distm(cluster_parks[, c("Longitude", "Latitude")],
                               cluster_parks[, c("Longitude", "Latitude")],
                               fun = distHaversine))

    ## Solve the TSP problem using the nearest neighbor heuristic
  tsp_solution <- solve_TSP(TSP(dist_matrix), method = "nearest_insertion")
```

```
    ## Solution List
  tsp_results[[paste("Cluster", i)]] <- tsp_solution
}
```

Now that I have the results, I can visualize what the optimal route is. In order to do that, I want to plot them all on the same map which will require creating a function to compile all the data points into one table.

```
  ## Function to create segments from the TSP solution
get_tsp_segments <- function(cluster_number, top_parks_cluster, tsp_results) {

    ## Filter the parks by the selected cluster
  selected_cluster <- filter(top_parks_cluster, cluster == cluster_number)

    ## Retrieve the TSP order for the cluster
  tsp_order <- tsp_results[[paste("Cluster", cluster_number)]]
  tsp_solution_order <- TOUR(tsp_order)

    ## Order the parks according to the TSP solution
  ordered_cluster <- selected_cluster[tsp_solution_order, ]

    ## Close the loop by returning to the starting point
  ordered_cluster <- rbind(ordered_cluster, ordered_cluster[1, ])

    ## Create segments for each leg of the TSP path
  segments <- data.frame(cluster = cluster_number,
                         x = ordered_cluster$x[-nrow(ordered_cluster)],
                         y = ordered_cluster$y[-nrow(ordered_cluster)],
                         xend = ordered_cluster$x[-1],
                         yend = ordered_cluster$y[-1]
                         )

  return(segments)
}

  ## Call the function
all_segments <- lapply(1:max(top_parks_cluster$cluster), function(cluster_num) {
  get_tsp_segments(cluster_num, top_parks_cluster, tsp_results)
})

  ## Combine all segments into one data frame
all_segments_df <- do.call(rbind, all_segments)
```

Multi−park Trip Options