The primary purpose of this section is to introduce basic spatial analysis, visualization, and APIs in R. The secondary purpose is to examine the impact of state-level policy on the disposition of farmers' markets in the Southwest. As in a previous section, we'll pull data from data.gov on 7,863 farmers' markets in the United States.[1] Save the file as farmers-mkts.csv in the data sub-directory and plot the locations:

```
library(maps)
data <- read.csv("../data/farmers-mkts.csv", header = TRUE)
map("state", interior = FALSE)
map("state", boundary = FALSE, col = "gray", add = TRUE)
points(data$x, data$y, cex = 0.2, col = "blue")
```

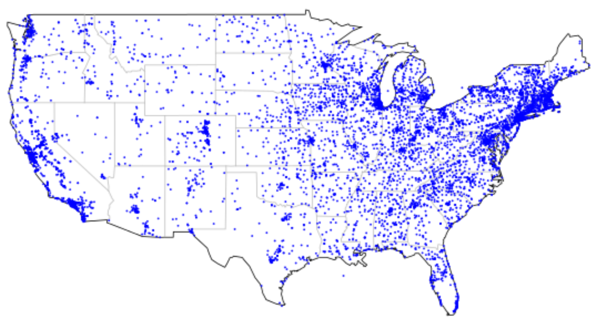

Figure 1: **US farmers markets**

Now consider only the farmers' markets in Colorado, Utah, New Mexico, and Arizona. There are 353 farmers markets in these four states. (One is mislabeled, and is actually in Pennsylvania. Knock this out.)

```
statelist <- c("New Mexico", "Colorado", "Arizona", "Utah")
state.data <- data[data$State %in% statelist, ]
state.data <- state.data[state.data$x < -80,]
dim(state.data)
```

```
[1] 353  32
```

Each column of the state.data frame contains information on a different feature of the market. The last 24 columns are binary variables with entries "Y" or "N", indicating whether the market sells cheese, for example, or accepts credit cards. Is it possible to predict the state of farmers' markets, purely based on these features? If so, then there may be something about state policy that has a direct, observable impact on the composition of the markets. Clean up the features, assigning a numerical indicator to the "Y" response:

```
X <- state.data[, 8:ncol(state.data)]
X <- apply(X, 2, function(col) { ifelse(col == "Y", 1, 0) })
colnames(X)[-(1:13)]
```

```
 [1] "Honey"    "Jams"     "Maple"    "Meat"     "Nursery"  "Nuts"
 [7] "Plants"   "Poultry"  "Prepared" "Soap"     "Trees"    "Wine"
```

[1] https://explore.data.gov/d/wfna-38ey

Note that all variables in the feature matrix X are binary. There are a variety of standard econometric techniques to deal with multinomial data — but I am unaware of any that don't implicitly (or explicitly) rely on the euclidean metric to identify distances between the covariates. This constraint has a very strange impact on purely binary data, which we will examine in section. For now, though, create a distance matrix between the observations in X using the binary method to measure distance:

```
dist.mat <- dist(X, method = "binary")
```

The resulting matrix bound to dist.mat contains the distances between markets, computed as the proportion of *bits* that match among the rows of the feature matrix. This object is then the basis for the hierarchical clustering algorithm in R. Build and plot the tree.

```
hclust.res <- hclust(dist.mat)
cl <- cutree(hclust.res, k = 5)
plot(cut(as.dendrogram(hclust.res), h = 0)$upper, leaflab = "none")
```

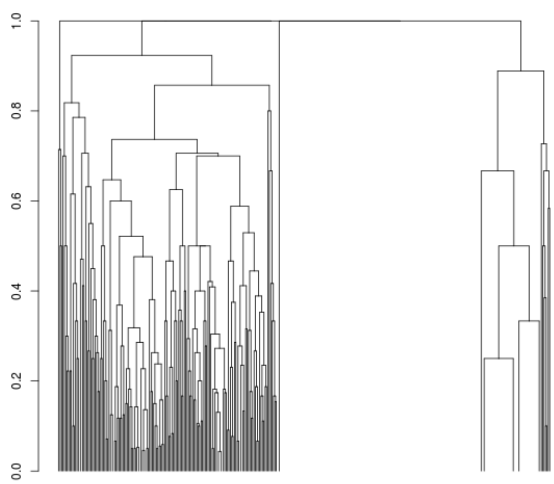

Figure 2: **Dendrogram**

The dendrogram of the hierarchical clustering is plotted in Figure 2. Cut the tree so that there are five branches. One branch only has three markets, which according to the data sell nothing at all. Throw this branch out and plot the markets again, but only for the four sample states. Figure 3 the points by cluster and highlighting the cluster that seems to indicate New Mexico.

```
coords <- state.data[ , c("x", "y")]

assignColor <- function(cl.idx) {
  col.codes <- c("#FF8000", "#0080FF", "#FFBF00", "#FF4000")
  return(col.codes[cl.idx])
}

map("state", interior = FALSE,
    xlim = c(-117, -101), ylim = c(28, 43))
map("state", boundary = FALSE, col="gray", add = TRUE,
    xlim = c(-117, -101), ylim = c(28, 43))
points(coords[["x"]], coords[["y"]], cex = 1, pch = 20, col = assignColor(cl))
```

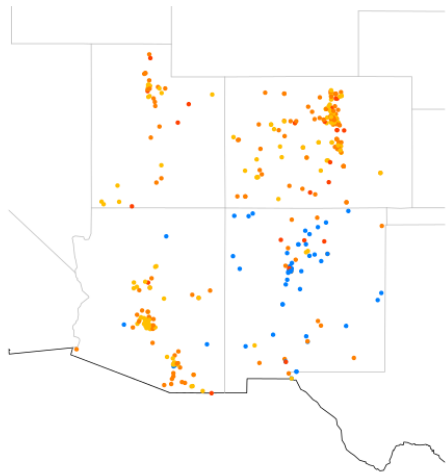

Figure 3: **Classified markets for the Southwest**

It is reasonably clear from these diagrams that farmers' markets in New Mexico are distinctive, somehow. But we can force the analysis into a more traditional regression discontinuity test. First, calculate the distance of each market to the New Mexico border between Arizona and Colorado. I have plugged in the three points to define `segment`, the simple, upside-down and backwards L-shaped border.

```
segment <- cbind(c(-109.047546, -109.047546, -103.002319),
                 c(31.33487100, 36.99816600, 36.99816600))

.segDistance <- function(coord) {
  near.obj <- nearestPointOnSegment(segment, coord)
  return(as.numeric(near.obj[["distance"]]))
}
```

The local function `.segDistance` will return the distance between the supplied coordinate to the global line segment. Apply this function to all coordinates. The resulting object `dist` represents distance to the New Mexico border; and to indicate the side of the border, scale the distance for each market *within* New Mexico by -1. A distance of zero is the border. This is beginning to look more and more like the regression discontinuity design, with the discontinuity at zero distance.

```
dist <- apply(coords, 1, FUN = .segDistance)
dist <- dist * ifelse(state.data[["State"]] == "New Mexico", -1, 1)
```

Now, plot the predicted cluster with respect to distance from border. The result is plotted in Figure 4 and indicates a clear discontinuity. The regression discontinuity analysis that we learn is generally for functions, not correspondences.

```
sel.cl <- cl < 5
plot(dist[sel.cl], cl[sel.cl], pch = 20, col = "blue",
     xlab = "Distance to New Mexico border (in degrees)",
     ylab = "Cluster category", yaxt = "n")
abline(v = 0, lty = 3, col = "red")
axis(2, at = 1:4)
```

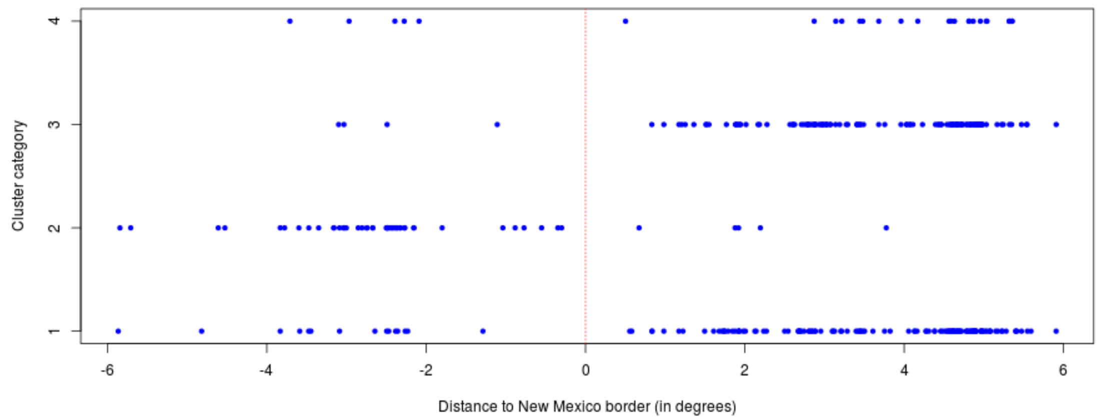

Figure 4: **Discontinuity at the border**

The graph in Figure 4 is not hemicontinuous, indicating some discontinuity. All the figures combined offer reasonably strong evidence that the New Mexico border significantly alters the composition of the markets.

Switch gears. Suppose, now, that we want to find the elevations of each of the farmers' markets in the four sample states. For this, we can use the Google Elevation API, which relies on URL requests, like we've seen in previous sections. The following two functions build the URL request for a collectino of coordinates.

```
convertCoords <- function(coord.collection) {
  apply(coord.collection, 1, function(x) { paste(x[2], x[1], sep = ",") })
}

getElevation <- function(coord.collection) {
  base.url <- "http://maps.googleapis.com/maps/api/elevation/json?locations="
  params <- "&sensor=false"
  coord.str <- paste(convertCoords(coord.collection), collapse = "|")
  query <- paste(base.url, coord.str, params, sep="")
  gotten <- getURL(query)

  output <- fromJSON(gotten, unexpected.escape = "skip")$results

  .elev <- function(x) {
    return(x[1][["elevation"]])
  }

  res <- as.matrix(lapply(output, .elev))
  return(res)
}
```

The Google API does not accept URLs that are too long. I am not sure what qualifies as too long, but the 353 farmers' market coordinates throw an error. So, we'll partition the coordinate collection.

```
partition <- function(df, each = 10) {
  s <- seq(ceiling(nrow(df) / each))
  suppressWarnings(res <- split(df, rep(s, each = each)))
  return(res)
}
```

```r
elev.split <- lapply(partition(coords), getElevation)
elevation <- unlist(elev.split)
```

Applying the `getElevation` function to each partition will send out multiple requests. The `elevation` collection contains the elevation for all farmers' markets. This is pretty cool. We don't need to store the elevations on disk. We can rely on Google's data and raster sampling to grab the elevations on demand.

Almost done. The maps in `R` are decent. But they are static and difficult to explore. Instead, use CartoDB to view and explore the data, uploading directly from `R`. Adjust the account name and API key accordingly.

```r
library(CartoDB)
cartodb("danhammer", api.key = "... paste your (my) api key here ...")
```

You will need to log into the CartoDB console and create a table with the appropriately named columns. I'll show you how to sign up for a free account and set up a table in section. Call this table `markets`. The following functions will send the coordinates, elevations, and cluster identifiers to the `markets` table.

```r
uploadMarket <- function(record, table.name = "markets") {
  cartodb.row.insert(name = table.name,
                     columns = list("x", "y", "cluster", "elevation"),
                     values = as.list(record))
}

mkts <- data.frame(x = coords[["x"]], y = coords[["y"]],
                   cluster = cl, elevation = elevation)

apply(mkts, 1, uploadMarket)
```

Note that we don't need to assign the output to a variable; the side effect is the upload of each row in `mkts` to the `markets` CartoDB table. (Again, we'll go over this in section.) Once the data are in CartoDB, we have access to a host of incredible visualization tools. You can even share the map:

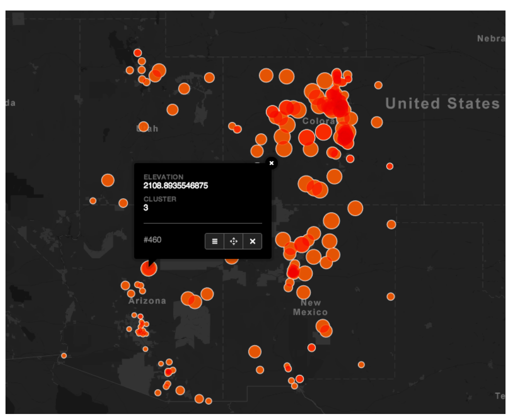

Figure 5: cdb.io/ZWfkdw

You can, along with your co-authors, explore the spatial data in an open and collaborative way. The size of the dots, here, indicate the elevation. Colorado markets are at higher elevation than the other three states; but the elevation is similar for markets just on either side of the New Mexico border. Elevation, then, may be a good cofactor to use in the regression discontinuity analysis. Next time.