Carson Keeter

Homework 1

Note: Full code is listed at the end

Also note: It's best to view this in its HTML version. That can be found here (https://drive.google.com/drive/folders/1K4nMjfwxWwSCjny9EMBk3-BBTRXmNCUC?usp=sharing). Just download the HTML file and you're set.

Question 1: Work with the dataset that containsaverage temperatures in and around the state of Colorado (colorados-t.dat). In this file, temperature is measured in 10s of degrees Celsius. For this question we are working with January temps (column name Jan). (24 points)

a) Read in the datasetand make a spatial (sfclass) object with the appropriate UTM projection. (2 points)

First, the appropriate libraries are read in:

```
library(maps)                                   # Draws geographical maps
library(maptools)                               # Manipulates maps
library(sf)                                     # Simple features
library(tmap)                                   # More maps
library(mapproj)                                # Projects maps
library(gstat)                                  # Geostatistical modeling
library(tools)                                  # General tools for R
library(dplyr)                                  # Data wranglin'
library(stringr)                                # Character wranglin'
library(tidyr)                                  # Data cleanin'
library(rgeos)                                  # Open source geometry
library(spdep)                                  # More geostatistical modeling
library(ggplot2)                                # Pretty plots
```

Then, the .dat file is read in using the following code:

```
colorado_dat_0 <- read.delim("colorados-t.dat")          # Raw .dat file

colorado_dat <- st_as_sf(
  x = colorado_dat_0,
  coords = c(
    "Lon",
    "Lat"
    ),
  crs = CRS(
    "+proj=utm +zone=13"
    )
  )
```

Additionally, some initial data manipulation is done to make mapping counties, states, and the point data slightly easier:

```
latlong2county2 <- function(pointsDF) {                    # Function to find state, county value
  county <- map('county', fill=TRUE, col='transparent', plot=FALSE)

  IDs <- sapply(strsplit(county$names, ':'), function(x) x[1])

  county_sp <- map2SpatialPolygons(
    county,
    IDs=IDs,
    proj4string=CRS('+proj=longlat +datum=WGS84')
  )

  pointsSP <- SpatialPoints(
    pointsDF,

    proj4string=CRS('+proj=longlat +datum=WGS84'))

  indices <- over(pointsSP, county_sp)

  countyNames <- sapply(county_sp@polygons, function(x) x@ID)

  countyNames[indices]

}

colorado_dat$county <- latlong2county2(data.frame(colorado_dat_0$Lon, colorado_dat_0$Lat))  # Cr
eates county variable

colorado_dat[, 16:17] <- str_split_fixed(colorado_dat$county, pattern = ",", n = 2)         # Se
parates county and state

names(colorado_dat)[16:17]<-c("state", "county")            # Renamin'

colorado_dat[, 15] <- NULL                                  # Cleanin'

colorado_dat <- colorado_dat %>%                            # Separates county and state
  mutate(state = toTitleCase(state),                        # Capitalizes characters within colu
mn
         county = toTitleCase(county)
         ) %>%
  mutate(name = paste0(county, sep = ", ", state)
         )

for(i in colorado_dat$county){                     # Replaces duplicate county names with NA for
mapping purposes

  colorado_dat$county[duplicated(colorado_dat$county)] <- NA

}
```

Lastly, a dataframe containing arbitrary points within each state is created to allow for easier reading of the final map:

```
state_point <- data.frame(
  state = c("Wyoming", "Nebraska", "Kansas", "Oklahoma", "New Mexico", "Colorado", "Utah"),
  Lon = c(-105.861625, -102.086648, -101.310184, -102.072771, -108.434457,-102.624180, -109.5626
68),
  Lat = c(41.3, 41.256259, 39.0, 36.745762, 36.180285, 40.853806, 39.187828)
)

state_point <- st_as_sf(
  x = state_point,
  coords = c(
    "Lon",
    "Lat"
  ),
  crs = CRS(
    "+proj=utm +zone=13"
  )
)
```

For now, using *UTM* projection seems the most appropriate. The area in which we are working is far from the poles and is relatively small. Additionally, most of the data falls into UTM zone 13.

b) Provide a choropleth map of observed January temperature sat the observed locations, overlaid over the borders of Colorado counties. (Hint: use the mapspackage to get Colorado county borders. Use sensible choices for all map aesthetics and appropriate labels to ensure maximal readability). (6points)

First, we need to define the area in which we're working. Based on the coordinate data, it seems that most of the data fall in and around the state of Colorado.

The following code creates a baselayer map with the relevant states:

```
state_fill <- map(
  database = 'state',
  regions = c(
    "Colorado",
    "Wyoming",
    "New Mexico",
    "Utah",
    "Kansas",
    "Nebraska",
    "Oklahoma",
    "Texas",
    "Arizona"
  ),
  fill = T,
  plot = F
)
```

Then, this code creates a baselayer map for the counties within each state:

```
states_map <- map(
  database = 'county',
  regions = c(
    "Colorado",
    "Wyoming",
    "New Mexico",
    "Utah",
    "Kansas",
    "Nebraska",
    "Oklahoma",
    "Texas",
    "Arizona"
  ),
  fill = T,
  plot = F
)
```

Then, these maps are converted to *spatial polygons* so they can be properly visualized:

```
states_poly <- map2SpatialPolygons(
  states_map,
  IDs = states_map$names,
  proj4string = CRS(
    "+proj=utm +zone=13"
    )
)

states_fill_poly <- map2SpatialPolygons(
  state_fill,
  IDs = state_fill$names,
  proj4string = CRS(
    "+proj=utm +zone=13"
  )
)
```

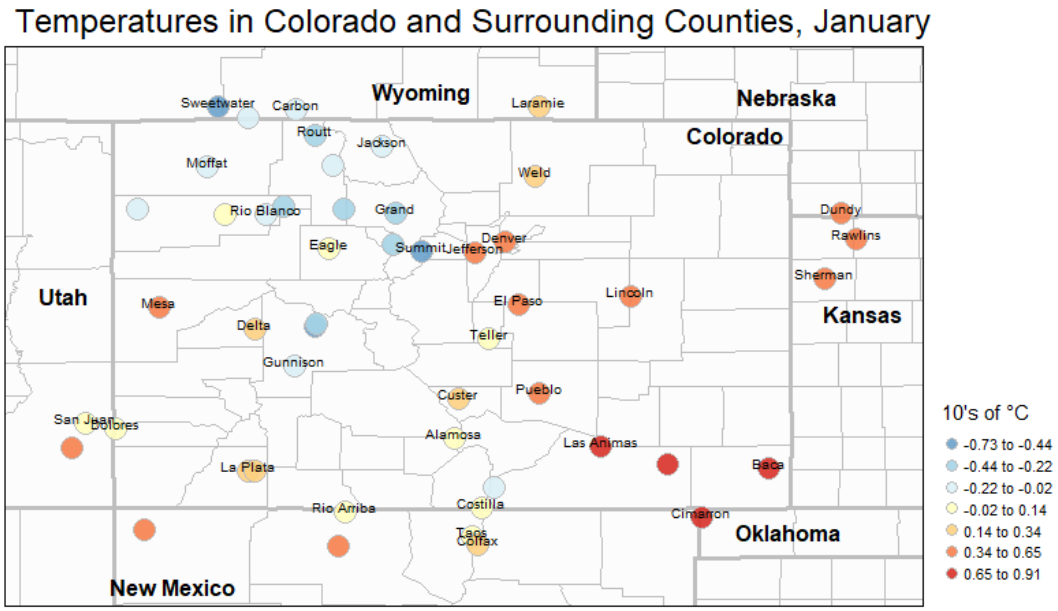
Once the maps have been converted to *spatial polygons*, the boundry of the map must be determined. In my mind, it seems easiest to add some abritrary value to the maximum and minimum latitude and longitude, rather than fiddle with the limits manually:

```
max_lon <- max(colorado_dat_0$Lon)
min_lon <- min(colorado_dat_0$Lon)

max_lat <- max(colorado_dat_0$Lat)
min_lat <- min(colorado_dat_0$Lat)
```

Once those are determined, we can create a chloropleth map of observed January temperatures from these locations. Comments can be found within the margins of the following code:

```
tm_shape(
  states_fill_poly,                          # Defines the first base layer (state borders)
  ylim=c(
    min_lat - .5,                            # Adds half a degree to the figure width/height
    max_lat + .5
  ),
  xlim=c(
    min_lon - .5,
    max_lon + .5
  )
) +
  tm_fill(
    col = "gray99"
  ) +
  tm_borders(
    lwd = 3,
    col = "gray"
  )  +
tm_shape(                                    # Defines county borders
  states_poly,
  ylim=c(
    min_lat - .5,
    max_lat + .5
    ),
  xlim=c(
    min_lon - .5,
    max_lon + .5
    )
) +
  tm_borders(
    lwd = 1.5,
    col = 'gray'
  ) +
  tm_shape(                                  # Plots temperature data points
    colorado_dat
    ) +
  tm_symbols(
    col = "Jan",
    palette = "-RdYlBu",
    n = 7,
    style = "jenks",
    border.lwd = 0.2,
    border.col = 'gray',
    alpha = 0.9,
    scale = 1.15,
    title.col = "10's of °C"
  ) +
  tm_text(                                   # Adds county labels
    "county",
    textNA = "",
    size = .65,
    just = "bottom"
  ) +
```

```
tm_shape(                                    # Adds final layer of state labels
  state_point
) +
tm_text(
  "state",
  textNA = "",
  remove.overlap = F,
  shadow = T,
  fontface = "bold"

) +
tm_legend(
  position = c(
    "left",
    "bottom"
    ),
  legend.outside = TRUE,
  frame = F,
  main.title = 'Temperatures in Colorado and Surrounding Counties, January'
)
```



The figure above represents the average temperature measured at variance weather stations in and around the state of Colorado. One can notice cooler (at and below freezing) temperatures clustered in the northwest and southwest of the state, and warmer temperatures towards the east. This discrepancy is most likely due to high elevation (i.e. Rocky Mountain Range).

c) Estimate and plot a classic binned omni-directional (isotropic) variogram using January temperatures, specifying a constant mean. Use sensible options with respect to

bin size and distance cutoff. (4 points)

Variograms track differences in data between different locations. To create one, a few parameters must be specified first:

```
cutoff <- .5*max(
  dist(
    cbind(
      colorado_dat_0$Lon,
      colorado_dat_0$Lat
      )
    )
  )

bins <- 30
```

Cutoff specifies the maximum distance for which to compute the variogram. I have chosen 30 bins and half the maximum distance as that seemed to come out with the most reasonable variogram (read: after about half an hour of tinkering with different values).

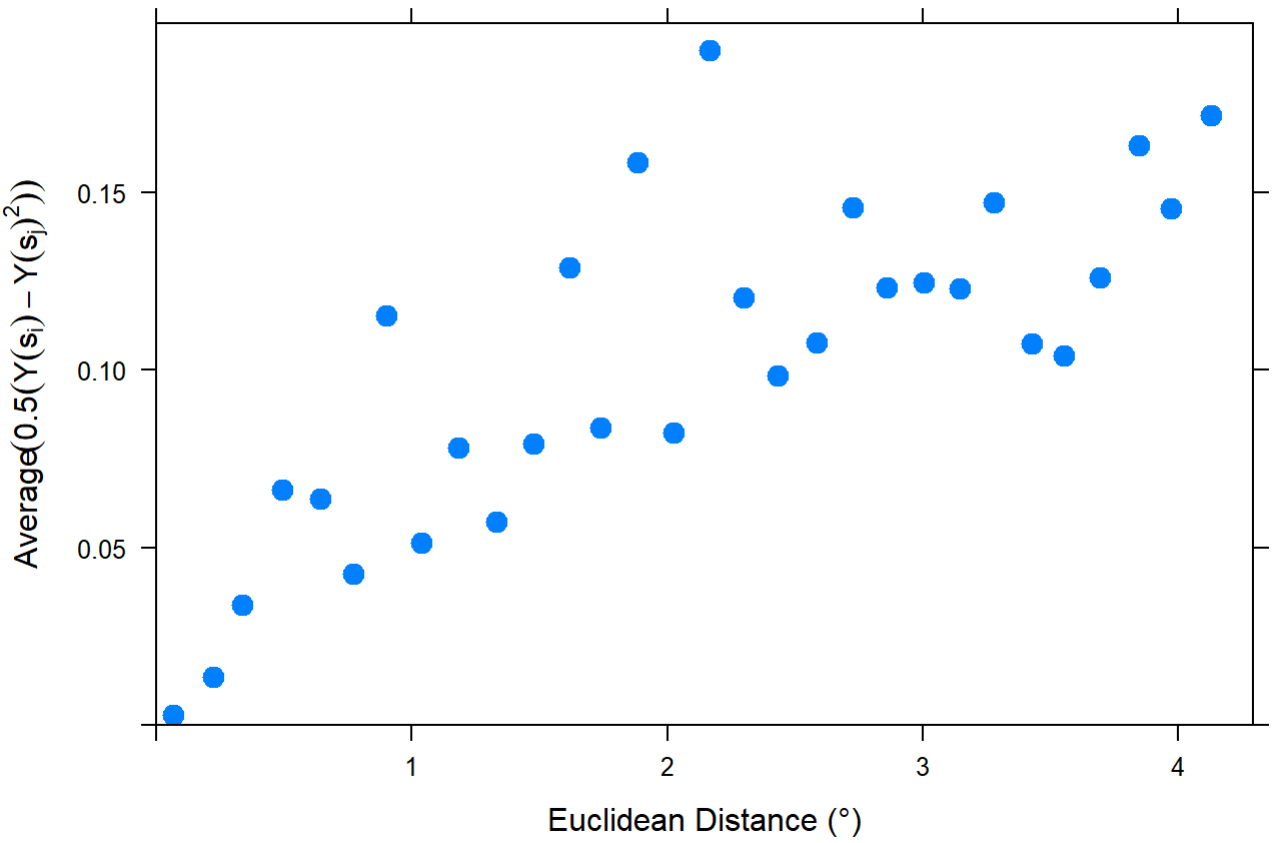
Then, we are able to compute the variogram with the following code:

```
variogram <- variogram(
  Jan ~ 1,                                  # 1 implies no covariates and a constant mean
  locations = ~Lat + Lon,
  data = colorado_dat_0,
  cutoff = cutoff,
  width = cutoff/bins
)

head(variogram)
```

```
##   np       dist       gamma dir.hor dir.ver   id
## 1  3 0.06664379 0.002766667       0       0 var1
## 2  2 0.22093446 0.013525000       0       0 var1
## 3 13 0.33746949 0.033746154       0       0 var1
## 4 14 0.49090538 0.066110714       0       0 var1
## 5 19 0.64019641 0.063694737       0       0 var1
## 6 22 0.77004457 0.042702273       0       0 var1
```

With these data, we can then plot the empircal variogram:

```
plot(
  x = variogram,
  ylab = expression(
    paste(
      "Average", (0.5*(Y(s[i]) - Y(s[j])^2))
      )
    ),
  xlab = "Euclidean Distance (°)",
  cex = 2,
  pch = 20,
  main = "Omnidirection Variogram of Temps. in CO and Surrounding Counties, Jan"
  )
```



d) Based on your answer from part c), identify approximate values for the following: 1) nugget; 2) range; 3) sill; 4) percent nugget. (4 points; 1 point each)

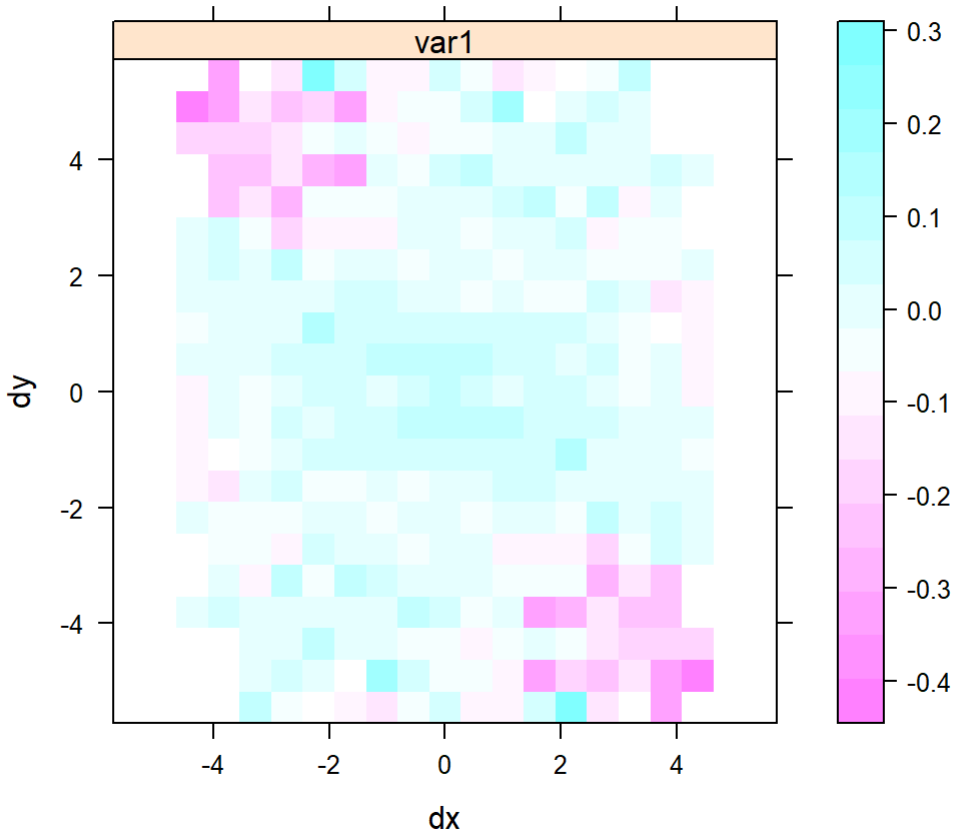
Solely based on visual estimates, the *nugget* appears to be very close to 0 (and a *% nugget* close to 0%), the *sill* is approx. 0.12, with a *range* of 2.5°.

e) Investigate January temperatures for evidence of anisotropy using any tools you deem appropriate. Provide graphical evidence and summarize your approach and what you found. (6 points)

To find evidence of aniostropy, a co-variogram map will be constructed. Co-variogram maps are able to reveal spatial dependence as it is not omni-directional. And since Colorado is nearly a perfect square, this should accurately reveal where covariance is different.

The following code constructs the relevant structures to create the co-variogram map:

```
cutoff <- .65*max(
  dist(
    cbind(
      colorado_dat_0$Lon,
      colorado_dat_0$Lat
    )
  )
)

bins_co <- 10

covar_map <- variogram(                       # Uses same modelling syntax as variogram above
  Jan ~ 1,
  locations = ~Lat + Lon,
  data = colorado_dat_0,
  cutoff = cutoff,
  width = cutoff/bins_co,
  covariogram = TRUE,                         # Set = TRUE to include covariogram, rather than a
variogram
  map = TRUE
)

plot(
  x = covar_map,
  main = "Covariogram of Temps. in CO and Surrounding Counties, Jan"
     )
```



f) If you found evidence of anisotropy, make an informative guess as to why this is the case. If you did not, make an informative guess as to why not. (2 points)

From the covariogram above, there is noticeable spatial dependence mostly in the *northwest-southeast* direction. In these pink areas, spatial correlation is negative. This means that there is reason to believe that these data are *anisotropic*.

Question 2: For this question, work with the US County shapefile used in Activity 1. US County shp.zip is already uploaded under Activity 1 on WyoCourses. We are going to focus on just the counties in Texas. (18 points)

a) Read in the shapefile into R. What is the default projection listed? What spatial feature does this projection preserve for mapping purposes? (3 points)

To read in the .shp file, the following code is used:

```
US_map <- st_read("acs_county_us.shp")
```

```
## Reading layer `acs_county_us' from data source `H:\Desktop\Spatial\Homework 1\acs_county_us.s
hp' using driver `ESRI Shapefile'
## Simple feature collection with 3141 features and 5 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -2031905 ymin: -2427680 xmax: 2516374 ymax: 732103.3
## epsg (SRID):    NA
## proj4string:    +proj=laea +lat_0=45 +lon_0=-100 +x_0=0 +y_0=0 +a=6370997 +b=6370997 +units=m
+no_defs
```

This .shp of the US is using *Lambert equal area* projection which correctly shows relative sizes of areas. This type of projection is appropriate for relatively small distances such as counties and states.

b) Make a subset of the US file that only contains counties in Texas and make a choropleth map with two panels, one showing % uninsured by county in 2012, the other showing % uninsured by county in 2016, with a common legend. Make sensible choices about map aesthetics to ensure maximal readability. (6 points)

Once the data is read in, a subset of data only containing *Texas* is created:

```
US_map <- st_transform(                                    # Creates dataframe and sf object
  x = US_map,
  crs = "+proj=laea +lat_0=45 +lon_0=-100"
)

US_map$state<-sub(".*, ", "", US_map$NAME)                 # Separates "Name" column by "State"

texas <- US_map %>%                                        # Creates a new dataframe only containi
ng Texas
  filter(
    state == "Texas"
    )
```

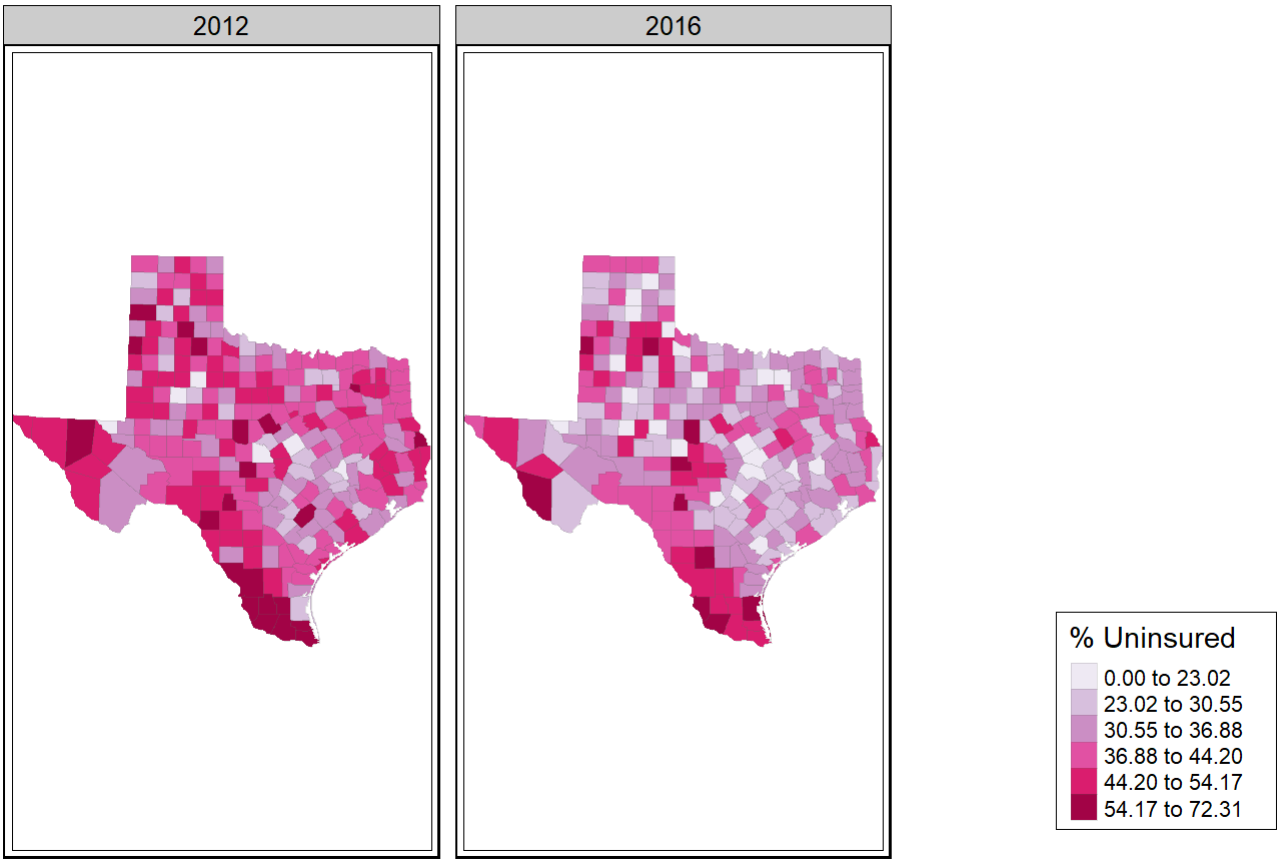
Since the purpose of this exercise is to create two chloropleth maps separated by year with a common legend, it is also necessary to create a “stacked” version of the data as well:

```
texas_gather <- gather(
  data = texas,
  key = "year",
  value = "perc_un",
  un_2012:un_2016
)

texas_gather <- texas_gather %>%                          # This looks fancy but it just renames t
he variables
  mutate(
    year = ifelse(
      test = year == "un_2012",
      yes = "2012",
      no = "2016"
      )
    )
```

This essentially allows for easier plotting and faceting. The data now contains one column of all the uninsurance values and another column with the respective year. The information is still preserved, just shaped differently.

Now, a chloropleth map can be created:

```
uninsured_tx <- tm_shape(
  shp = texas_gather                                    # Data with polygon and uninsurance data
  ) +
  tm_polygons(
    col = "perc_un",                                    # Colors counties based on uninsurance r
ates
    palette = 'PuRd',
    n = 6,
    style = "fisher",
    title = "% Uninsured",
    border.lwd = 0,
    border.alpha = 0.2
  ) +
  tm_legend(
    position = c("right", "bottom"),
    frame = T,
    outside = T,
    main.title = "% Uninsured in Texas, 2012-2016"
  ) +
  tm_facets(                                            # Separates maps based on year
    by = 'year'
  ) +
  tm_layout(
    bg.color = "white",
    frame.double.line = T,
    outer.bg.color = "white"
  )

uninsured_tx
```



The two maps correspond to the year in which the data was collected. There is a possible overall decrease in uninsurance rates from 2012 to 2016 (thanks, Obama). However, this is only by visually assessing the figures above. There are still some areas that seemed to have had no change or increased uninsurance rates. The next questions will determine if there is a true spatial correlation.

d) Using a “queen” polygon adjacency-based binary spatial structure, compute Global Moran’s I coefficients to determine whether the spatial correlation in % uninsured was stronger in 2012 or in 2016. Please justify your conclusion and show all work. (6 points)

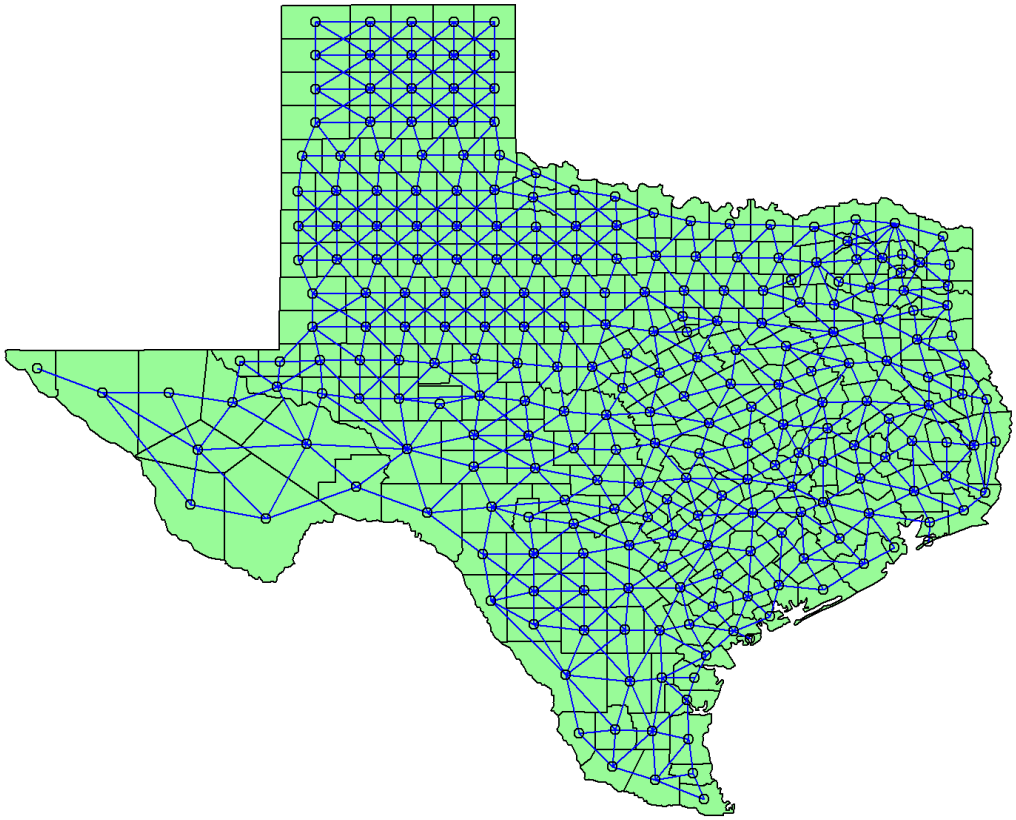
To create an adjacency-based binary spatial structure, maps containing counties in Texas are constructed:

```
tx_map <- map(
  'county',
  'texas',
  fill = T,
  plot = F
)

tx_poly <- st_geometry(texas)

tx_cent <- map2SpatialPolygons(
  tx_map,
  IDs = tx_map$names,
  proj4string = CRS(
    "+proj=laea +lat_0=45 +lon_0=-100"
    )
  )

tx_queen_1 <- poly2nb(
  tx_cent,
  queen = TRUE
)

tx_queen_2 <- poly2nb(
  tx_poly,
  queen = TRUE
)

centroids <- gCentroid(
  tx_cent,
  byid = TRUE
)
```

Once these data are wrangled, transformed and centroids are found, a map connecting each county to its neighbor can be constructed using the following:

```
plot(
  tx_cent,
  col = 'palegreen',
  main = "Queen Based Adjacency of Counties in Texas"
)
plot(
  tx_queen_1,
  coordinates(
    centroids
    ),
  col = "blue",
  add = TRUE
  )
```

Queen Based Adjacency of Counties in Texas



This figure shows each county and the number of neighbors it has using the Queen Based Adjacency method. If a county has 1 or more points of contact with another county, they are considered neighbors. Additionally, this method considered regions neighbors if they have one corner touching, rather than a large border (the southwest counties, for example).

Now, to compute Moran's Global I for each year:

```
moran_2012 <- moran.test(
  x = texas$un_2012,
  listw = nb2listw(
    neighbours = tx_queen_2,
    style = "B"
    )
  )

moran_2016 <- moran.test(
  x = texas$un_2016,
  listw = nb2listw(
    neighbours = tx_queen_2,
    style = "B"
  )
)

moran_2012
```

```
##
##  Moran I test under randomisation
##
## data:  texas$un_2012
## weights: nb2listw(neighbours = tx_queen_2, style = "B")
##
## Moran I statistic standard deviate = 4.6375, p-value = 1.763e-06
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.166036767      -0.003952569       0.001343602
```

```
moran_2016
```

```
##
##  Moran I test under randomisation
##
## data:  texas$un_2016
## weights: nb2listw(neighbours = tx_queen_2, style = "B")
##
## Moran I statistic standard deviate = 7.3757, p-value = 8.173e-14
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.266036426      -0.003952569       0.001339932
```

Since *spatial correlation* is the statistic in question, a dataframe containing only the estimates is created to better see the differences between years:

```
comp_df <- t(
  data.frame(
    "Uninsured 2012" = moran_2012$estimate,
    "Uninsured 2016" = moran_2016$estimate
    )
  )

comp_df
```

```
##                Moran I statistic  Expectation    Variance
## Uninsured.2012         0.1660368 -0.003952569 0.001343602
## Uninsured.2016         0.2660364 -0.003952569 0.001339932
```

Here, it is fairly obvious that the average spatial correlation, *Moran's I*, between neighbors regarding uninsurance rates in 2016 is higher than that of 2012. This means (possibly) that uninsurance rates were more wide-spread in 2012 and were concentrated in certain areas in 2016.

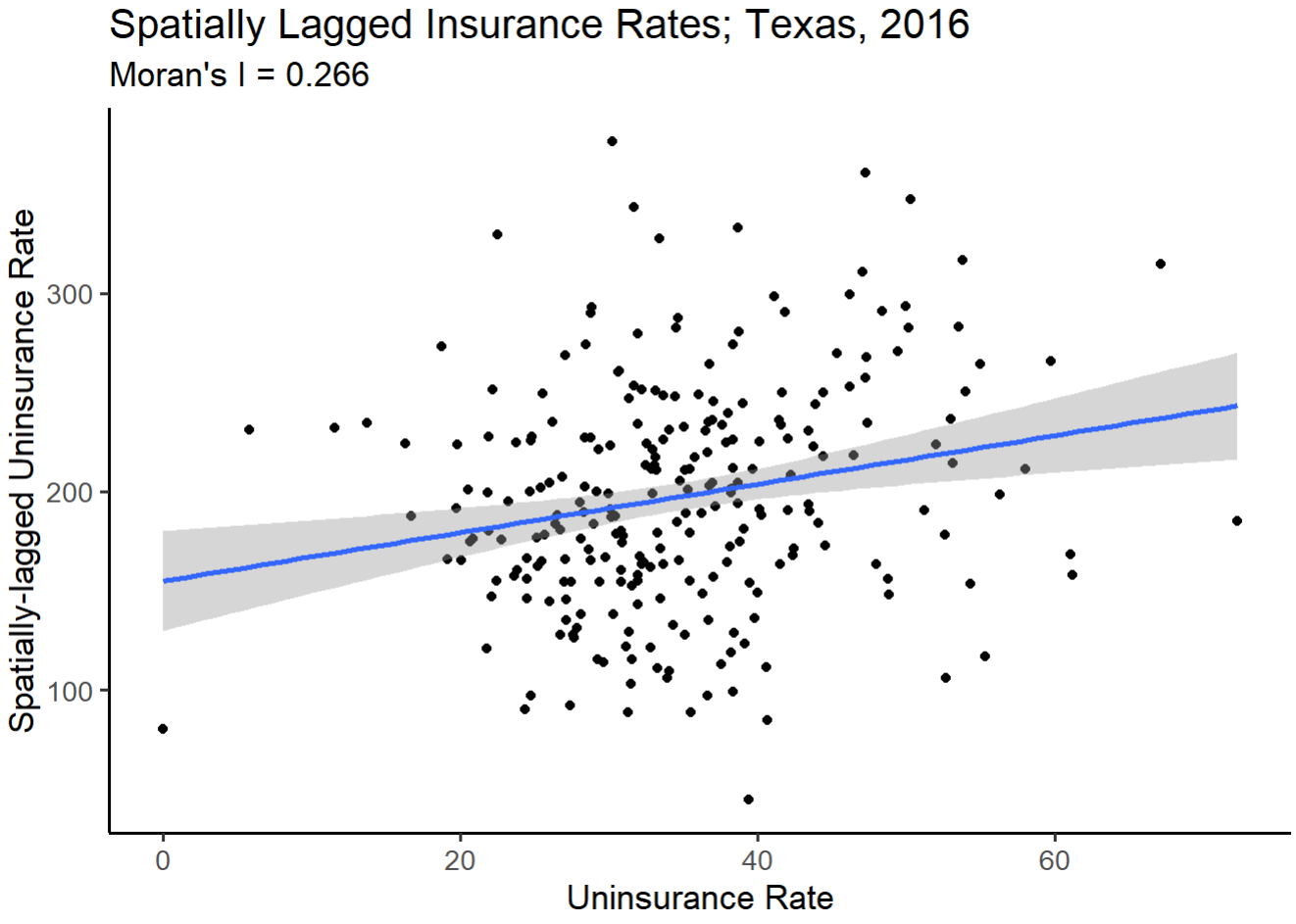
e) Compute and plot the Moran's I correlogram for % uninsured in 2016 using the same spatial structure as in part c). Provide a complete interpretation of your figure. (3 points)

Using the same structures from above, we first pre-multiply the adjacency matrix (a binary matrix using 1's to denote neighboring counties and 0 for non-neighboring counties) by the 2016 uninsured data:

```
tx_for_lag <- nb2mat(
  neighbours = tx_queen_2,
  style = "B"
  )

lagged_2016 <-  tx_for_lag %*% texas$un_2016
```

Then, that data can be used to construct a figure using spatially lagged data to visualize *Moran's I* (i.e. spatial correlation).

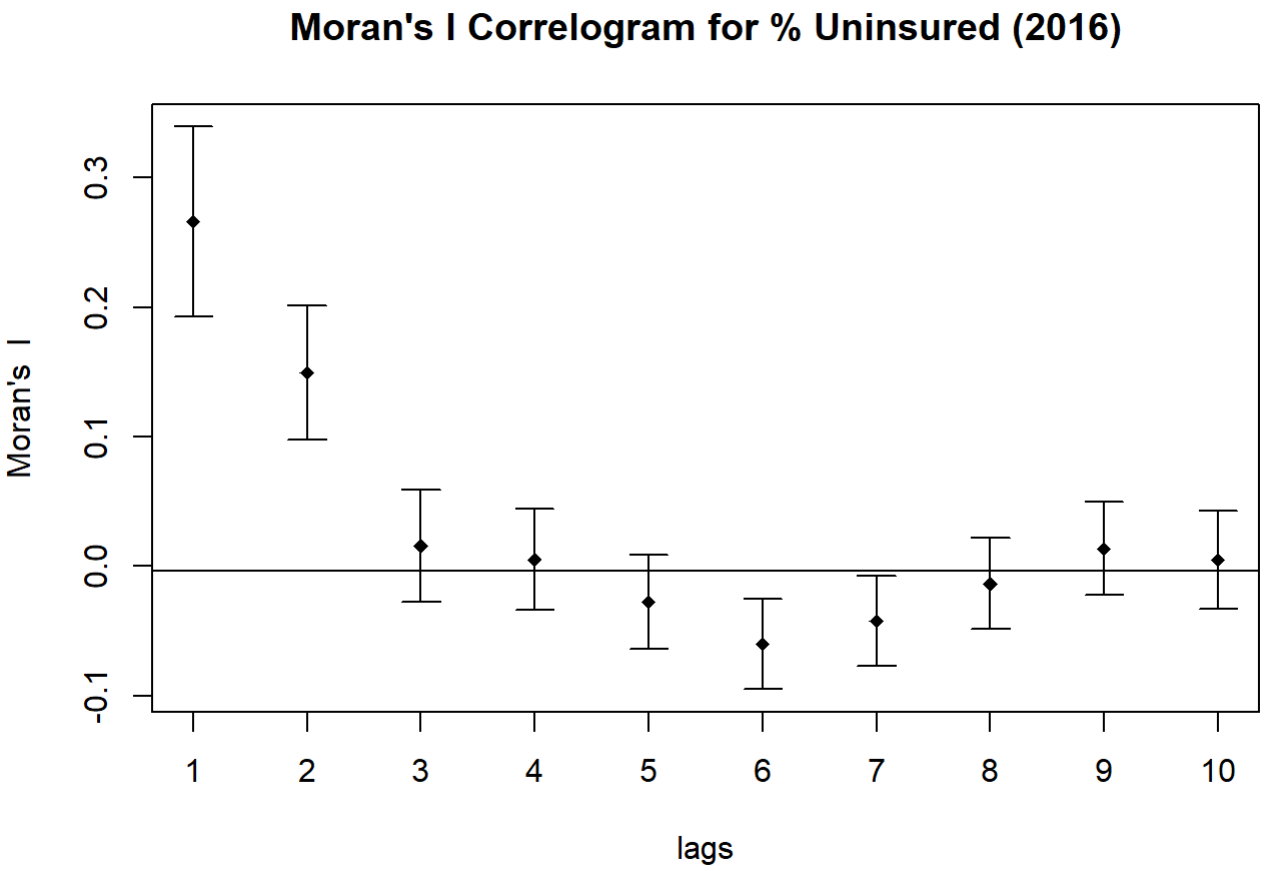
```
ggplot(                                          # Dataframe containing uninsurance data
  data = texas,
  aes(
    x = un_2016,
    y = lagged_2016
  )
) +
  geom_point(                                    # Plots points

  ) +
  geom_smooth(
    method = "lm"                                # Creates regression line using a 'linear
model'
  ) +
  theme_classic(
    base_size = 13
  ) +
  labs(
    x = "Uninsurance Rate",
    y = "Spatially-lagged Uninsurance Rate",
    title = "Spatially Lagged Insurance Rates; Texas, 2016",
    subtitle = "Moran's I = 0.266"
  )
```



As one can see, the spread of these data around the regression line mimic an R of approx. 0.266.

Now, to actually visualize *spatial correlation* with regard to the number of neighbors:

```
plot(
  sp.correlogram(
    tx_queen_2,
    texas$un_2016,
    order = 10,
    style = "B",
    method = "I"
    ),
  main = "Moran's I Correlogram for % Uninsured (2016)"
  )
```



This figure shows the relationship between *spatial correlation (Moran's I)* and the number of neighbors considered (lags). We see, therefore, that spatial correlation is the highest when one neighbor is considered with regard to uninsurance rates. As more neighbors are considerd, spatial correlation decreases.

Full Code

The code below is the entirety of the analysis. You may find some plots and calculations here that were not reported above (mostly due to many plots expressing the same information). If you run this on your own, you'll experience only a fraction of the torment I experienecd. You'll also see all of the warnings, messages, etc that I've hidden in the markdown above.

```r
setwd("H:/Desktop/Spatial/Homework 1")

### Spatial Statistics: Homework 1 ###


# Libraries needed

library(maps)
library(maptools)
library(sf)
library(tmap)
library(tmaptools)
library(mapproj)
library(gstat)
library(tools)
library(dplyr)
library(stringr)
library(tidyr)
library(rgeos)
library(spdep)
library(ggplot2)

################################################################################ Question 1 ####
######################
# Load and read in data

colorado_dat_0 <- read.delim("colorados-t.dat")                                  # Raw .dat file

latlong2county2 <- function(pointsDF) {                                           # Function to find
state value

  county <- map('county', fill=TRUE, col='transparent', plot=FALSE)

  IDs <- sapply(strsplit(county$names, ':'), function(x) x[1])

  county_sp <- map2SpatialPolygons(
    county,
    IDs=IDs,
    proj4string=CRS('+proj=longlat +datum=WGS84')
  )

  pointsSP <- SpatialPoints(
    pointsDF,
    proj4string=CRS('+proj=longlat +datum=WGS84'))

  indices <- over(pointsSP, county_sp)

  countyNames <- sapply(county_sp@polygons, function(x) x@ID)

  countyNames[indices]

}
```

```
names(colorado_dat_0)
```

```
##  [1] "Lon"       "Lat"       "Elevation" "Jan"       "Feb"
##  [6] "Mar"       "Apr"       "May"       "Jun"       "Jul"
## [11] "Aug"       "Sep"       "Oct"       "Nov"       "Dec"
```

```
colorado_dat <- st_as_sf(
  x = colorado_dat_0,
  coords = c(
    "Lon",
    "Lat"
    ),
  crs = CRS(
    "+proj=utm +zone=13"
    )
  )

colorado_dat$county <- latlong2county2(data.frame(colorado_dat_0$Lon, colorado_dat_0$Lat))

colorado_dat[, 16:17] <- str_split_fixed(colorado_dat$county ,pattern = ",", n = 2)

names(colorado_dat)[16:17]<-c("state", "county")

colorado_dat[, 15] <- NULL

state_point <- data.frame(
  state = c("Wyoming", "Nebraska", "Kansas", "Oklahoma", "New Mexico", "Colorado", "Utah"),
  Lon = c(-105.861625, -102.086648, -101.310184, -102.072771, -108.434457,-102.624180, -109.5626
68),
  Lat = c(41.3, 41.256259, 39.0, 36.745762, 36.180285, 40.853806, 39.187828)
)

state_point <- st_as_sf(
  x = state_point,
  coords = c(
    "Lon",
    "Lat"
  ),
  crs = CRS(
    "+proj=utm +zone=13"
  )
)


colorado_dat <- colorado_dat %>%                                        # Separates county
 and state
  mutate(state = toTitleCase(state),
         county = toTitleCase(county)
         ) %>%
  mutate(name = paste0(county, sep = ", ", state)
         )

for(i in colorado_dat$county){                                          # Replaces duplicat
e county names with NA
  colorado_dat$county[duplicated(colorado_dat$county)] <- NA
}

table(colorado_dat$county)
```

```
##
##    Alamosa       Baca     Carbon   Cimarron     Colfax   Costilla
##          1          1          1          1          1          1
##     Custer      Delta     Denver    Dolores      Dundy      Eagle
##          1          1          1          1          1          1
##    El Paso      Grand   Gunnison    Jackson  Jefferson   La Plata
##          1          1          1          1          1          1
##    Laramie Las Animas    Lincoln       Mesa     Moffat     Pueblo
##          1          1          1          1          1          1
##    Rawlins Rio Arriba Rio Blanco      Routt   San Juan    Sherman
##          1          1          1          1          1          1
##     Summit Sweetwater       Taos     Teller       Weld
##          1          1          1          1          1
```

```
# Create State Map

states_map <- map(
  database = 'county',
  regions = c(
    "Colorado",
    "Wyoming",
    "New Mexico",
    "Utah",
    "Kansas",
    "Nebraska",
    "Oklahoma",
    "Texas",
    "Arizona"
  ),
  fill = T,
  plot = F
)

state_fill <- map(
  database = 'state',
  regions = c(
    "Colorado",
    "Wyoming",
    "New Mexico",
    "Utah",
    "Kansas",
    "Nebraska",
    "Oklahoma",
    "Texas",
    "Arizona"
  ),
  fill = T,
  plot = F
)

states_poly <- map2SpatialPolygons(
  states_map,
  IDs = states_map$names,
  proj4string = CRS(
    "+proj=utm +zone=13"
    )
)

states_fill_poly <- map2SpatialPolygons(
  state_fill,
  IDs = state_fill$names,
  proj4string = CRS(
    "+proj=utm +zone=13"
  )
)

max_lon <- max(colorado_dat_0$Lon)
```

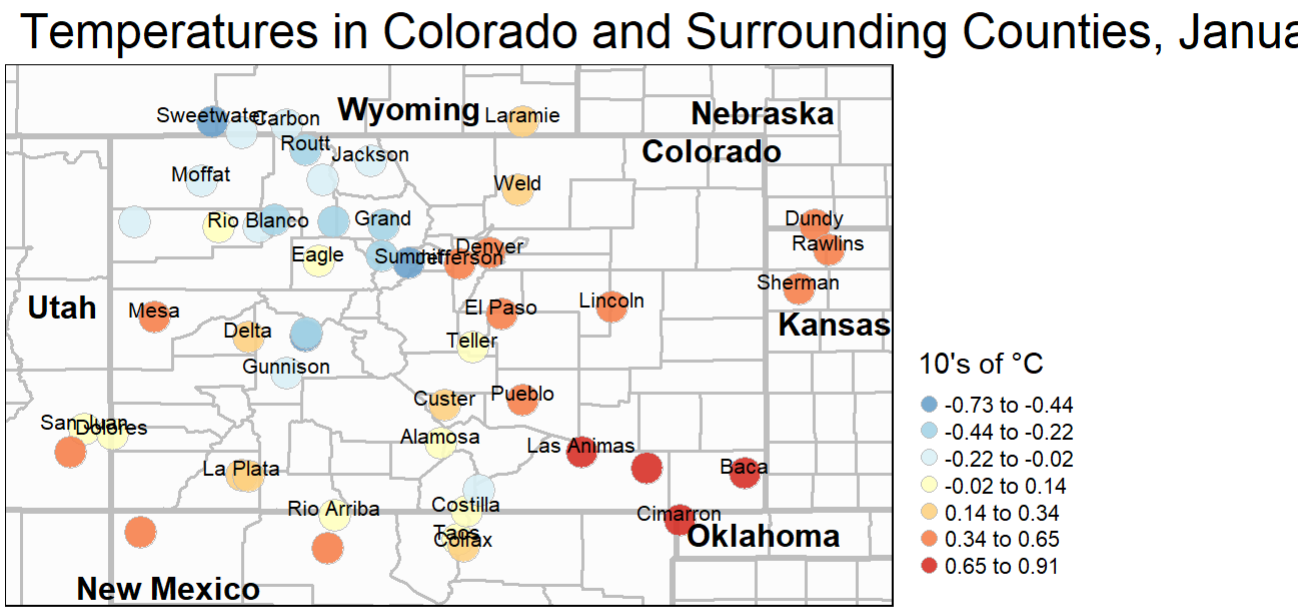
```
min_lon <- min(colorado_dat_0$Lon)

max_lat <- max(colorado_dat_0$Lat)
min_lat <- min(colorado_dat_0$Lat)

names(states_fill_poly)
```

```
## [1] "arizona"    "colorado"   "kansas"     "nebraska"   "new mexico"
## [6] "oklahoma"   "texas"      "utah"       "wyoming"
```

```
tm_shape(
  states_fill_poly,
  ylim=c(
    min_lat - .5,
    max_lat + .5
  ),
  xlim=c(
    min_lon - .5,
    max_lon + .5
  )
) +
  tm_fill(
    col = "gray99"
  ) +
  tm_borders(
    lwd = 3,
    col = "gray"
  )  +
tm_shape(
  states_poly,
  ylim=c(
    min_lat - .5,
    max_lat + .5
    ),
  xlim=c(
    min_lon - .5,
    max_lon + .5
    )
) +
  tm_borders(
    lwd = 1.5,
    col = 'gray'
  ) +
  tm_shape(
    colorado_dat
    ) +
  tm_symbols(
    col = "Jan",
    palette = "-RdYlBu",
    n = 7,
    style = "jenks",
    border.lwd = 0.2,
    border.col = 'gray',
    alpha = 0.9,
    scale = 1.15,
    title.col = "10's of °C"
  ) +
  tm_text(
    "county",
    textNA = "",
    size = .65,
    just = "bottom"
  ) +
```

```
  tm_shape(
    state_point
  ) +
  tm_text(
    "state",
    textNA = "",
    remove.overlap = F,
    shadow = T,
    fontface = "bold"

  ) +
  tm_legend(
    position = c(
      "left",
      "bottom"
      ),
    legend.outside = TRUE,
    frame = F,
    main.title = 'Temperatures in Colorado and Surrounding Counties, January'
  )
```

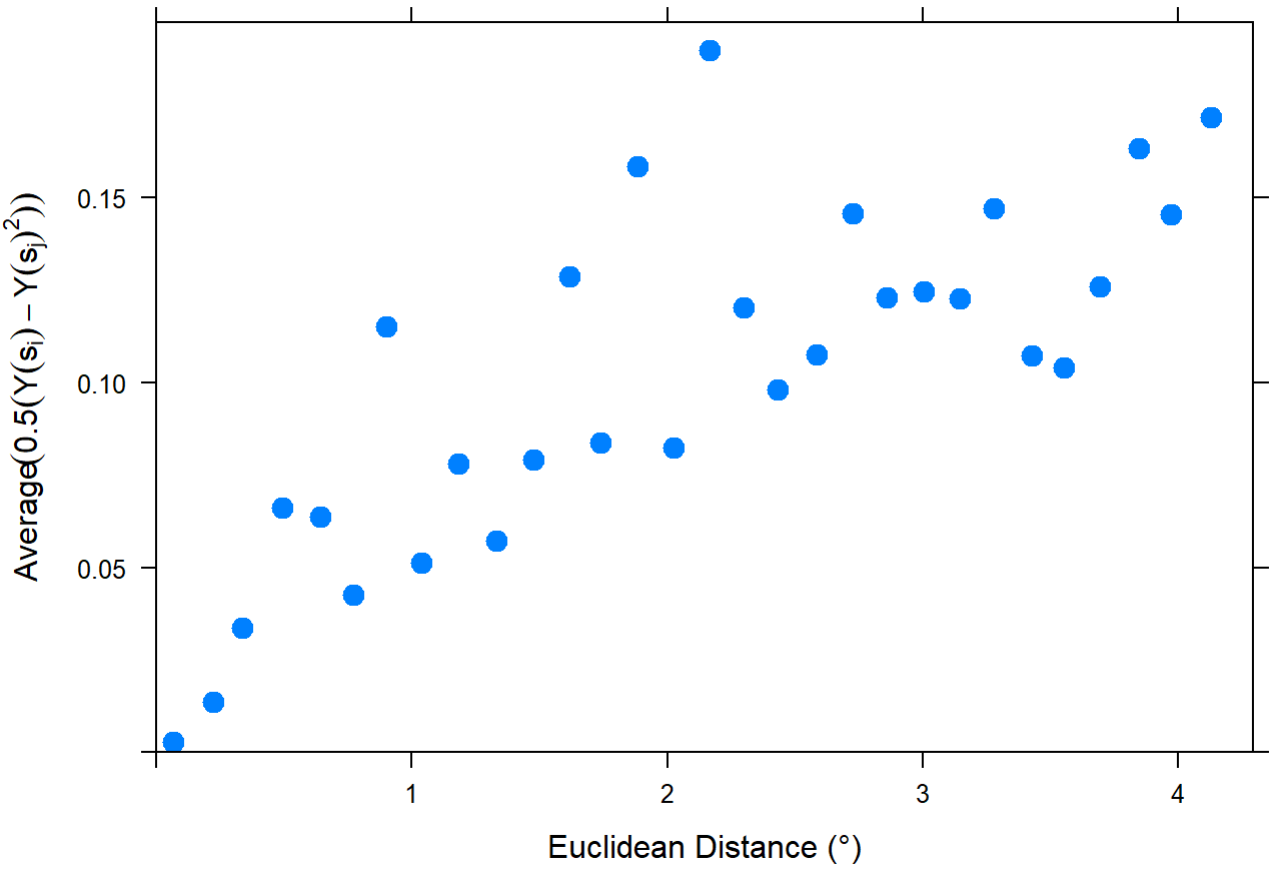
```
## Variable "Jan" contains positive and negative values, so midpoint is set to 0. Set midpoint = NA to show the full spectrum of the color palette.
```



```
cutoff <- .5*max(
  dist(
    cbind(
      colorado_dat_0$Lon,
      colorado_dat_0$Lat
      )
    )
  )

bins <- 30

variogram <- variogram(
  Jan ~ 1,
  locations = ~Lat + Lon,
  data = colorado_dat_0,
  cutoff = cutoff,
  width = cutoff/bins
)

variogram
```

```
##    np       dist       gamma dir.hor dir.ver   id
## 1   3 0.06664379 0.002766667       0       0 var1
## 2   2 0.22093446 0.013525000       0       0 var1
## 3  13 0.33746949 0.033746154       0       0 var1
## 4  14 0.49090538 0.066110714       0       0 var1
## 5  19 0.64019641 0.063694737       0       0 var1
## 6  22 0.77004457 0.042702273       0       0 var1
## 7  15 0.89857361 0.115246667       0       0 var1
## 8  23 1.03575308 0.051278261       0       0 var1
## 9  42 1.18330368 0.078035714       0       0 var1
## 10 28 1.33176443 0.057217857       0       0 var1
## 11 39 1.47423789 0.079166667       0       0 var1
## 12 41 1.61821229 0.128684146       0       0 var1
## 13 32 1.73983509 0.083642188       0       0 var1
## 14 28 1.88494819 0.158469643       0       0 var1
## 15 41 2.02529216 0.082269512       0       0 var1
## 16 32 2.16384921 0.189979688       0       0 var1
## 17 47 2.29752442 0.120304255       0       0 var1
## 18 46 2.43387682 0.098236957       0       0 var1
## 19 36 2.58343729 0.107608333       0       0 var1
## 20 41 2.72714522 0.145784146       0       0 var1
## 21 39 2.85925192 0.123161538       0       0 var1
## 22 35 3.00478133 0.124670000       0       0 var1
## 23 44 3.14413988 0.122775000       0       0 var1
## 24 40 3.27888538 0.147106250       0       0 var1
## 25 33 3.42642745 0.107348485       0       0 var1
## 26 42 3.55340165 0.104083333       0       0 var1
## 27 38 3.69418673 0.126068421       0       0 var1
## 28 37 3.84510684 0.163298649       0       0 var1
## 29 37 3.97264814 0.145408108       0       0 var1
## 30 25 4.12857231 0.171758000       0       0 var1
```

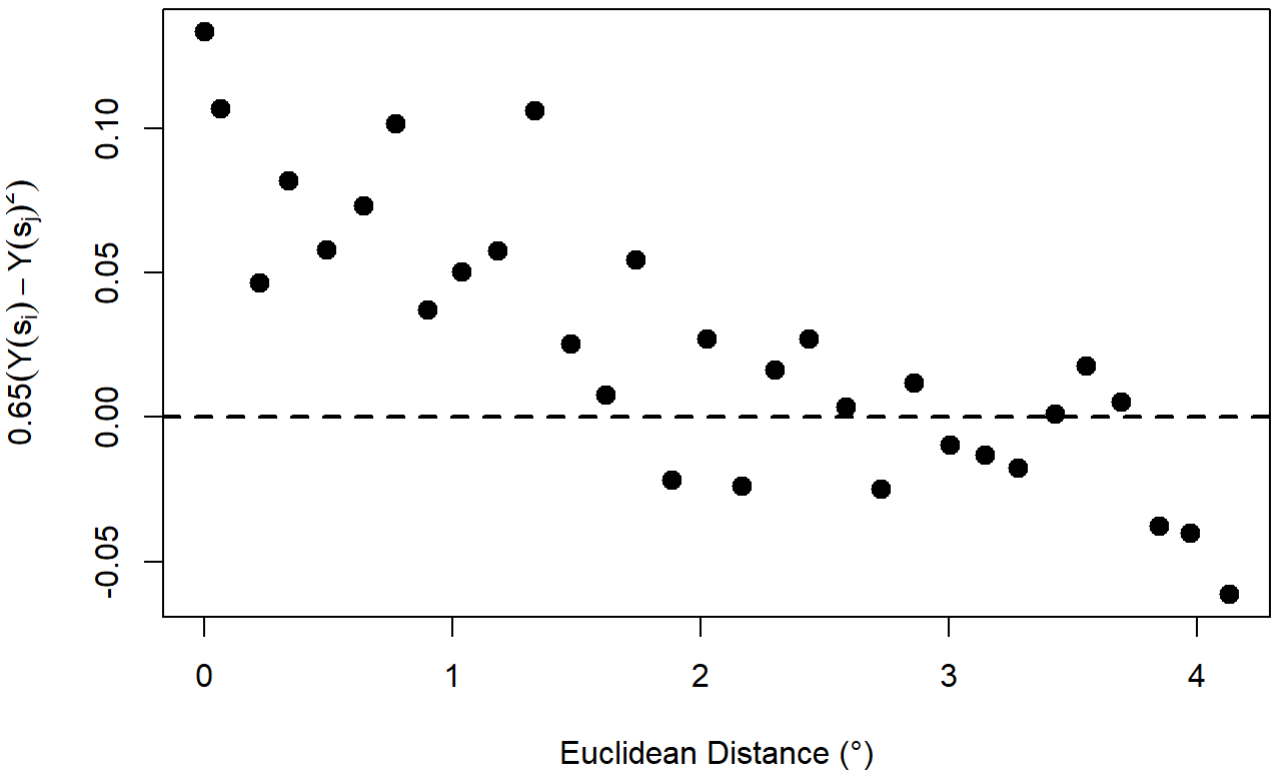
```
plot(
  variogram,
  ylab = expression(
    paste(
      "Average", (0.5*(Y(s[i]) - Y(s[j])^2))
      )
    ),
  xlab = "Euclidean Distance (°)",
  cex = 2,
  pch = 20
  )
```



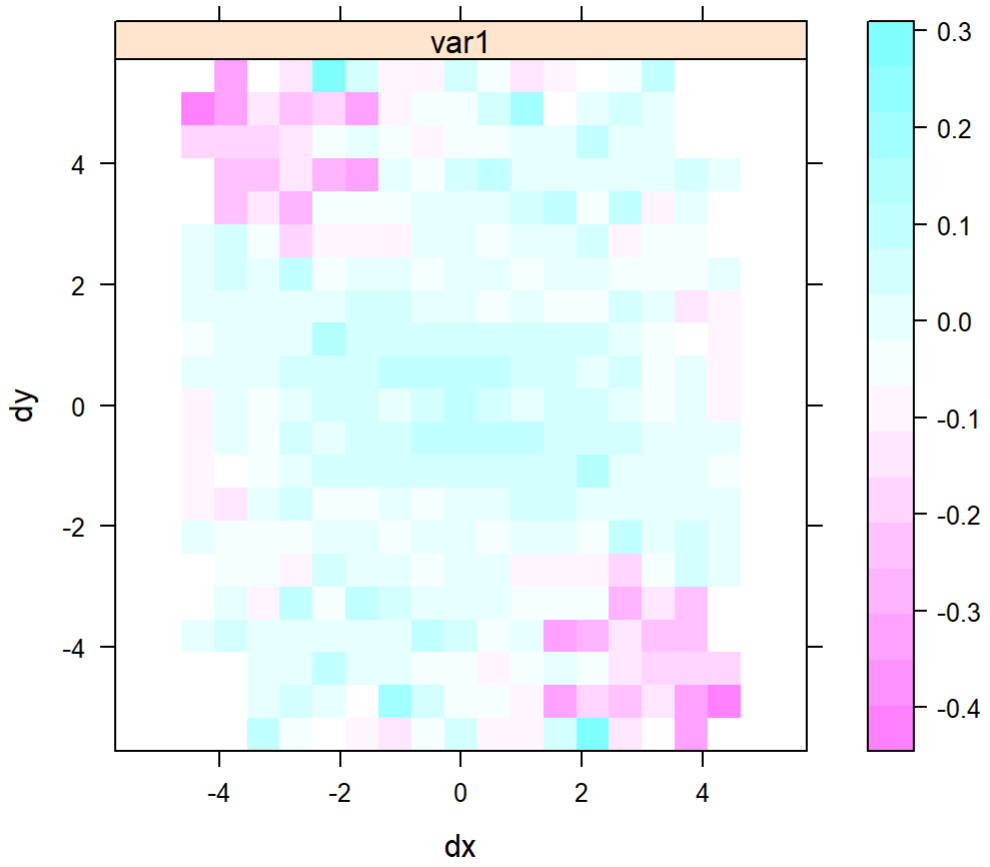

```
covariogram <- variogram(
  Jan ~ 1,
  locations = ~Lat + Lon,
  data = colorado_dat_0,
  cutoff = cutoff,
  width = cutoff/bins,
  covariogram = TRUE
)

covariogram
```

```
##    np       dist        gamma dir.hor dir.ver   id
## 1   3 0.06664379  0.106682667       0       0 var1
## 2   2 0.22093446  0.046476000       0       0 var1
## 3  13 0.33746949  0.081983692       0       0 var1
## 4  14 0.49090538  0.057898857       0       0 var1
## 5  19 0.64019641  0.073274947       0       0 var1
## 6  22 0.77004457  0.101585091       0       0 var1
## 7  15 0.89857361  0.037214667       0       0 var1
## 8  23 1.03575308  0.050373391       0       0 var1
## 9  42 1.18330368  0.057726000       0       0 var1
## 10 28 1.33176443  0.105977429       0       0 var1
## 11 39 1.47423789  0.025540615       0       0 var1
## 12 41 1.61821229  0.007746244       0       0 var1
## 13 32 1.73983509  0.054650375       0       0 var1
## 14 28 1.88494819 -0.021591143       0       0 var1
## 15 41 2.02529216  0.027105268       0       0 var1
## 16 32 2.16384921 -0.023721500       0       0 var1
## 17 47 2.29752442  0.016445362       0       0 var1
## 18 46 2.43387682  0.027240783       0       0 var1
## 19 36 2.58343729  0.003554889       0       0 var1
## 20 41 2.72714522 -0.024707415       0       0 var1
## 21 39 2.85925192  0.011824718       0       0 var1
## 22 35 3.00478133 -0.009560000       0       0 var1
## 23 44 3.14413988 -0.012932182       0       0 var1
## 24 40 3.27888538 -0.017325000       0       0 var1
## 25 33 3.42642745  0.001085697       0       0 var1
## 26 42 3.55340165  0.017871714       0       0 var1
## 27 38 3.69418673  0.005366526       0       0 var1
## 28 37 3.84510684 -0.037616973       0       0 var1
## 29 37 3.97264814 -0.040091568       0       0 var1
## 30 25 4.12857231 -0.061170400       0       0 var1
## 31 50 0.00000000  0.133300000       0       0 var1
```

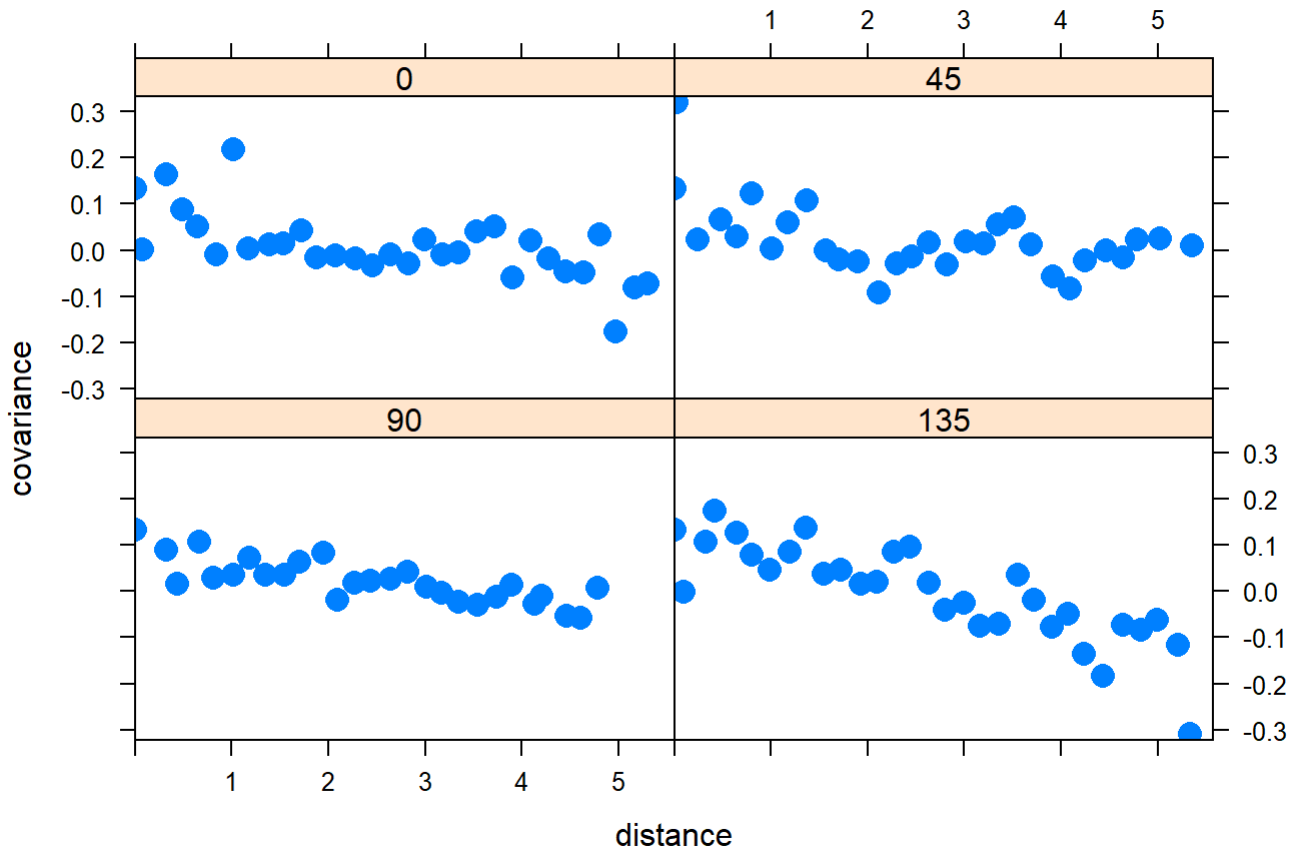
```
plot(
  gamma ~ dist,
  data = covariogram,
  ylab = expression(
    paste(
      0.65*(Y(s[i]) - Y(s[j])^2)
    )
  ),
  xlab = "Euclidean Distance (°)",
  cex = 2,
  pch = 20
)
abline(
  h = 0,
  lty = 2,
  col = 'black',
  lwd = 2
)
```




```
cutoff <- .65*max(
  dist(
    cbind(
      colorado_dat_0$Lon,
      colorado_dat_0$Lat
    )
  )
)

bins_co <- 10

covar_map <- variogram(
  Jan ~ 1,
  locations = ~Lat + Lon,
  data = colorado_dat_0,
  cutoff = cutoff,
  width = cutoff/bins_co,
  covariogram = TRUE,
  map = TRUE
)

plot(covar_map)
```




```
dir_covar <- variogram(
  Jan ~ 1,
  locations = ~Lat + Lon,
  data = colorado_dat_0,
  cutoff = cutoff,
  width = cutoff/bins,
  covariogram = TRUE,
  alpha = c(
    0,
    45,
    90,
    135
  )
)

plot(
  dir_covar,
  main = "Directional Covariograms",
  as.table = TRUE,
  cex = 1.5,
  pch = 16
)
```




```
########################################################################################## Questi
on 2 #################

US_map <- st_read("acs_county_us.shp")
```

```
## Reading layer `acs_county_us' from data source `H:\Desktop\Spatial\Homework 1\acs_county_us.s
hp' using driver `ESRI Shapefile'
## Simple feature collection with 3141 features and 5 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -2031905 ymin: -2427680 xmax: 2516374 ymax: 732103.3
## epsg (SRID):    NA
## proj4string:    +proj=laea +lat_0=45 +lon_0=-100 +x_0=0 +y_0=0 +a=6370997 +b=6370997 +units=m
+no_defs
```

```
class(US_map)
```

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

```
US_map <- st_transform(
  x = US_map,
  crs = "+proj=laea +lat_0=45 +lon_0=-100"
)

class(US_map)
```

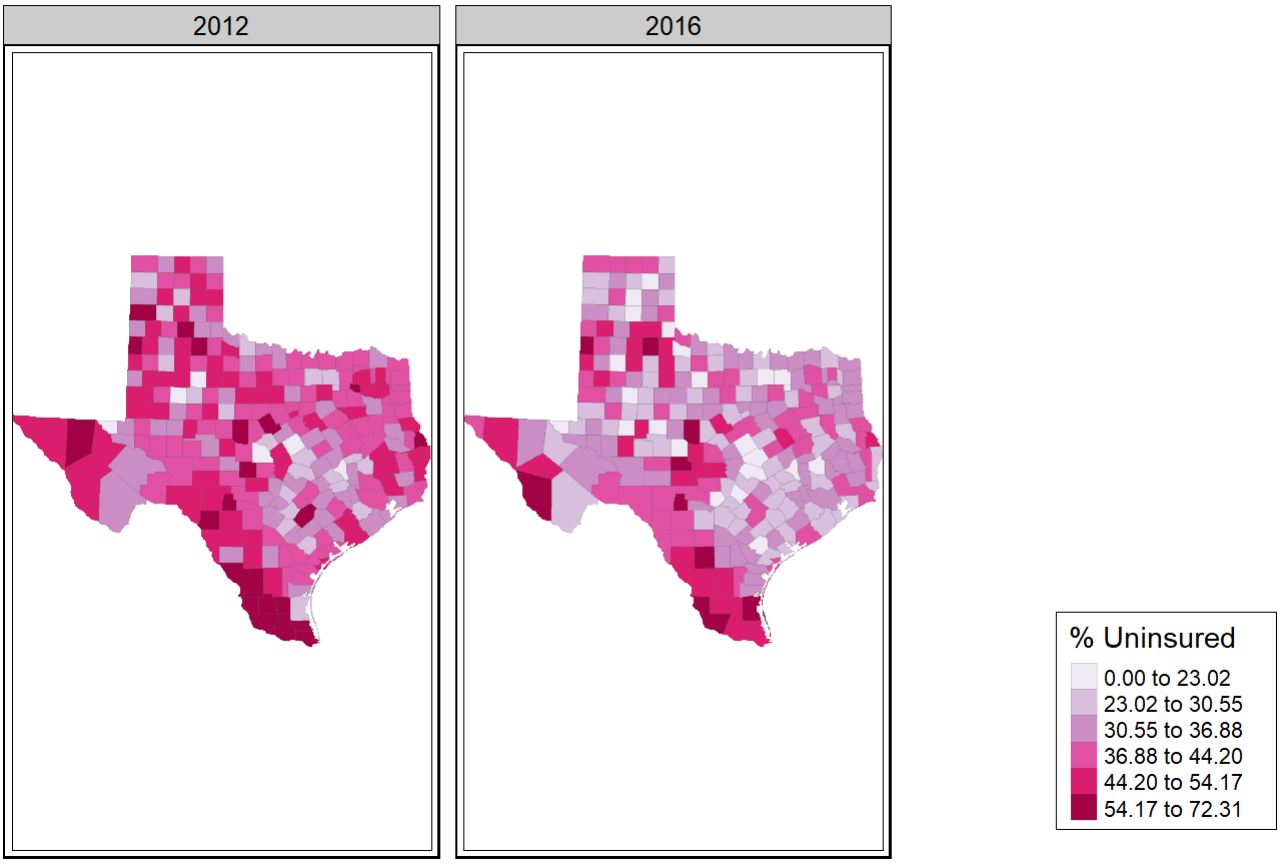
```
## [1] "sf"         "data.frame"
```

```
US_map$state<-sub(".*, ", "", US_map$NAME)

table(US_map$state)
```

```
##
##              Alabama               Alaska              Arizona
##                   67                   28                   15
##             Arkansas           California             Colorado
##                   75                   58                   64
##          Connecticut             Delaware District of Columbia
##                    8                    3                    1
##              Florida              Georgia               Hawaii
##                   67                  159                    5
##                Idaho             Illinois              Indiana
##                   44                  102                   92
##                 Iowa               Kansas             Kentucky
##                   99                  105                  120
##            Louisiana                Maine             Maryland
##                   64                   16                   24
##        Massachusetts             Michigan            Minnesota
##                   14                   83                   87
##          Mississippi             Missouri              Montana
##                   82                  115                   56
##             Nebraska               Nevada        New Hampshire
##                   93                   17                   10
##           New Jersey           New Mexico             New York
##                   21                   33                   62
##       North Carolina         North Dakota                 Ohio
##                  100                   53                   88
##             Oklahoma               Oregon         Pennsylvania
##                   77                   36                   67
##         Rhode Island       South Carolina         South Dakota
##                    5                   46                   66
##            Tennessee                Texas                 Utah
##                   95                  254                   29
##              Vermont             Virginia           Washington
##                   14                  133                   39
##        West Virginia            Wisconsin              Wyoming
##                   55                   72                   23
```

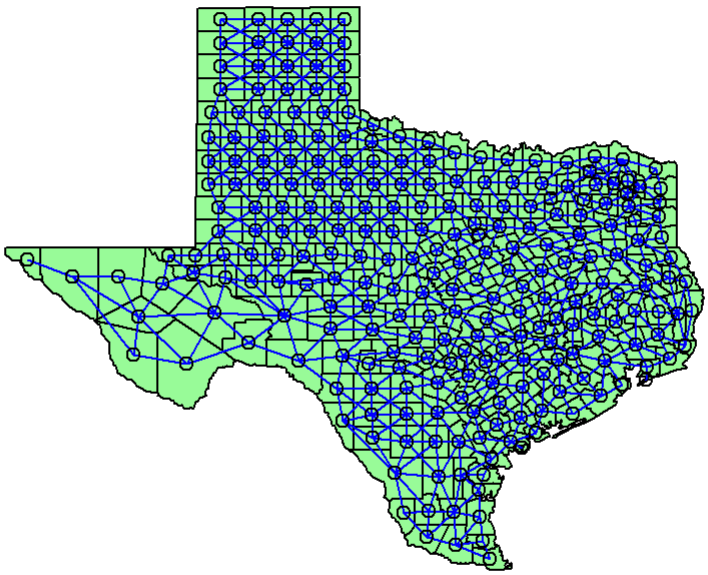
```
texas <- US_map %>%
  filter(
    state == "Texas"
    )

texas_gather <- gather(
  data = texas,
  key = "year",
  value = "perc_un",
  un_2012:un_2016
)

texas_gather <- texas_gather %>%
  mutate(
    year = ifelse(
      test = year == "un_2012",
      yes = "2012",
      no = "2016"
      )
    )

uninsured_tx <- tm_shape(
  shp = texas_gather
  ) +
  tm_polygons(
    col = "perc_un",
    palette = 'PuRd',
    n = 6,
    style = "fisher",
    title = "% Uninsured",
    border.lwd = 0,
    border.alpha = 0.2
  ) +
  tm_legend(
    position = c("right", "bottom"),
    frame = T,
    outside = T,
    main.title = "% Uninsured in Texas, 2012-2016"
  ) +
  tm_facets(
    by = 'year'
  ) +
  tm_layout(
    bg.color = "white",
    frame.double.line = T,
    outer.bg.color = "white"
  )

uninsured_tx
```

% Uninsured in Texas, 2012-2016



```
# Queen

tx_map <- map(
  'county',
  'texas',
  fill = T,
  plot = F
)

tx_poly <- st_geometry(texas)

tx_cent <- map2SpatialPolygons(
  tx_map,
  IDs = tx_map$names,
  proj4string = CRS(
    "+proj=laea +lat_0=45 +lon_0=-100"
    )
  )

tx_queen_1 <- poly2nb(
  tx_cent,
  queen = TRUE
)

tx_queen_2 <- poly2nb(
  tx_poly,
  queen = TRUE
)

centroids <- gCentroid(
  tx_cent,
  byid = TRUE
)

plot(
  tx_cent,
  col = 'palegreen',
  main = "Queen Based Adjacency of Counties in Texas"
)
plot(
  tx_queen_1,
  coordinates(
    centroids
    ),
  col = "blue",
  add = TRUE
  )
```

Queen Based Adjacency of Counties in Texas



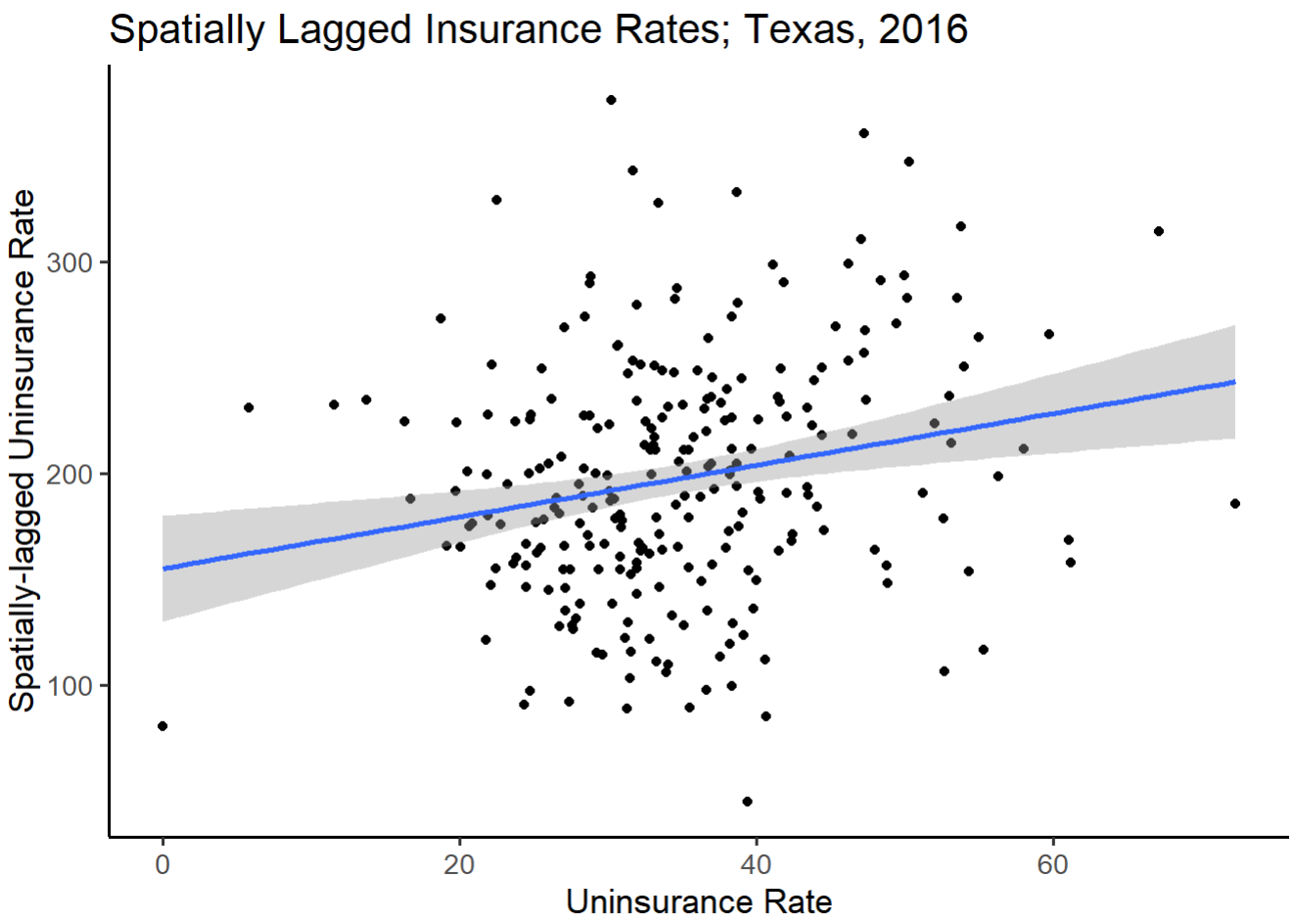
```
moran_2012 <- moran.test(
  x = texas$un_2012,
  listw = nb2listw(
    neighbours = tx_queen_2,
    style = "B"
    )
  )

moran_2012$statistic
```

```
## Moran I statistic standard deviate
##                           4.637525
```

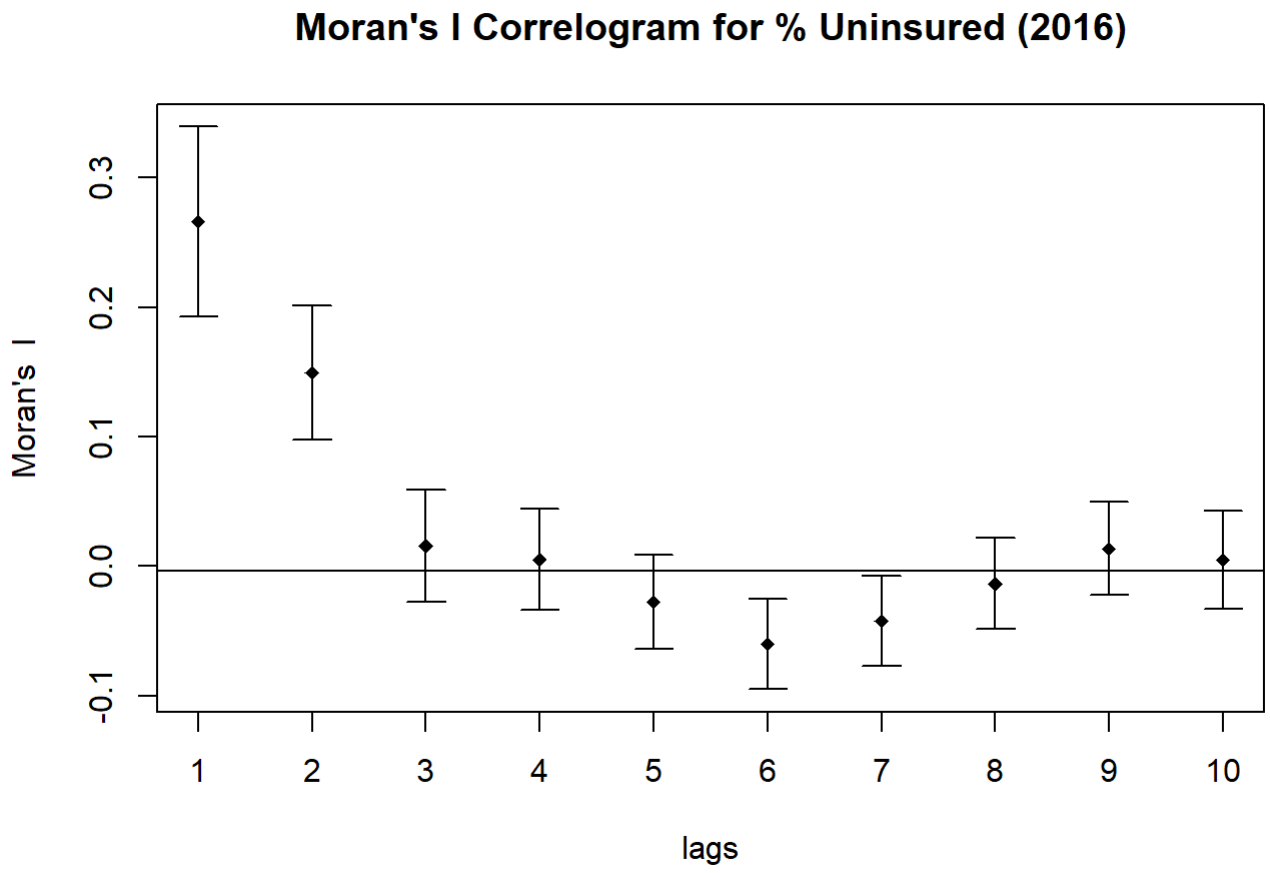
```
moran_2016 <- moran.test(
  x = texas$un_2016,
  listw = nb2listw(
    neighbours = tx_queen_2,
    style = "B"
  )
)

moran_2016
```

```
##
##  Moran I test under randomisation
##
## data:  texas$un_2016
## weights: nb2listw(neighbours = tx_queen_2, style = "B")
##
## Moran I statistic standard deviate = 7.3757, p-value = 8.173e-14
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.266036426      -0.003952569       0.001339932
```

```
comp_df <- t(
  data.frame(
    "Uninsured 2012" = moran_2012$estimate,
    "Uninsured 2016" = moran_2016$estimate
  )
)

tx_for_lag <- nb2mat(tx_queen_2, style = "B")

lagged_2016 <-  tx_for_lag %*% texas$un_2016

ggplot(
  data = texas,
  aes(
    x = un_2016,
    y = lagged_2016
  )
) +
  geom_point(

  ) +
  geom_smooth(
    method = "lm"
  ) +
  theme_classic(
    base_size = 13
  ) +
  labs(
    x = "Uninsurance Rate",
    y = "Spatially-lagged Uninsurance Rate",
    title = "Spatially Lagged Insurance Rates; Texas, 2016"
  )
```




```
plot(
  sp.correlogram(
    tx_queen_2,
    texas$un_2016,
    order = 10,
    style = "B",
    method = "I"
    ),
  main = "Moran's I Correlogram for % Uninsured (2016)"
  )
```


Moran's I Correlogram for % Uninsured (2016)
Moran's I
0.3
0.2
0.1
0.0
-0.1
1
2
3
4
5
6
7
8
9
10
lags

```
# The end
```


I HAVE NO IDEA
WHAT I'M DOING.
quickmeme.com