Yelp Reviews on Businesses Spatial Analysis

Brian Cervantes Alvarez | Polina Iasakova | Casey Nagai
March 15, 2025

Introduction

This project explores spatial patterns in Yelp businesses across various U.S. states, emphasizing the relationship between geographical location and customer perceptions, operational hours, and business characteristics. Our goal is to understand how spatial factors influence consumer ratings and business operations.

The dataset includes Yelp business data with star ratings (ranging from 1 to 5), location coordinates, credit card acceptance status, operating hours, and consumer sentiment indicators (cool, funny, useful). Due to dataset size and computational constraints, analyses were conducted separately for individual states, allowing detailed state-level insights.

Research Questions:

- Does spatial location influence consumer star ratings?
- How do star ratings correlate with operational factors such as credit card acceptance and weekly hours?
- What spatial patterns emerge from business operations across states?

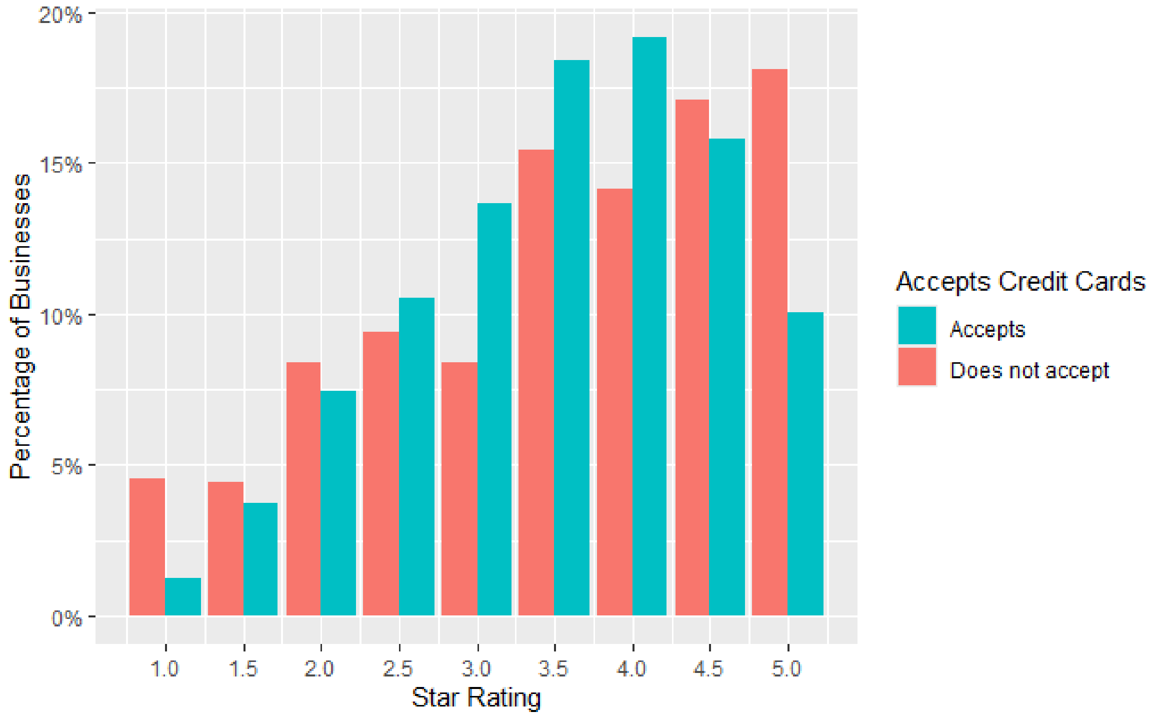
Methods

We employed exploratory data visualization techniques and spatial modeling approaches. Initially, we visualized star ratings relative to business acceptance of credit cards and their weekly operational hours. Spatial autocorrelation was then assessed using semivariograms to quantify the similarity of ratings across nearby locations. Variogram models, specifically the Matern model, were fitted, and spatial linear models (`splm`) were used to evaluate the significance of explanatory variables.

Results

Data Visualization

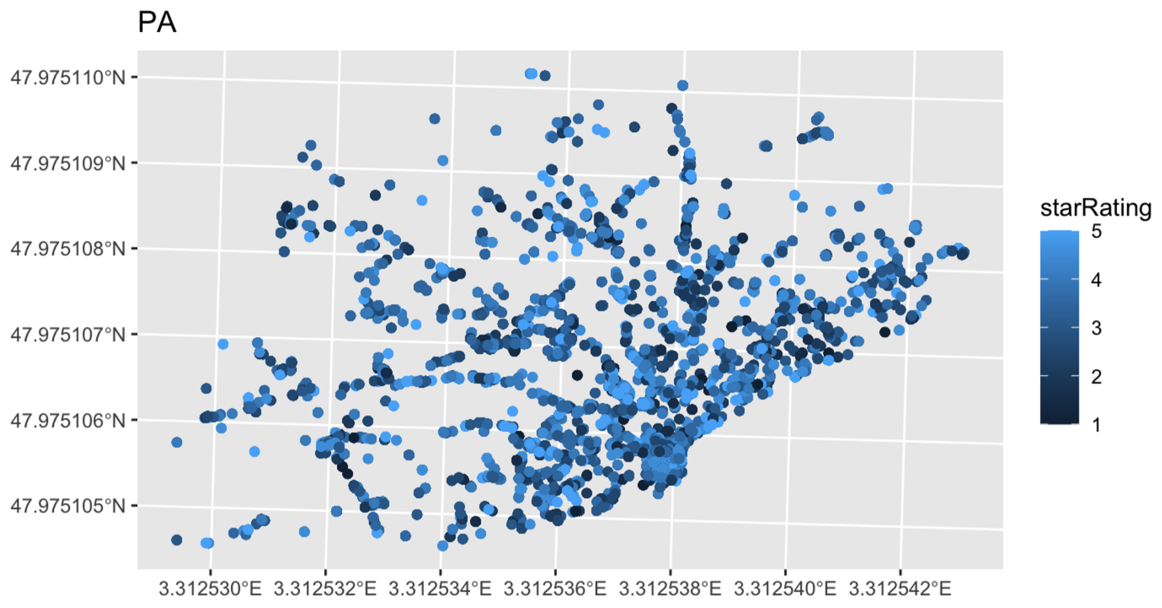
The visualization comparing star ratings against the acceptance of credit cards reveals distinct patterns (see Figure 1). Businesses accepting credit cards generally showed higher star ratings compared to those that did not, after normalization by the count of businesses accepting each payment type.



The weekly open-hour distribution indicated higher activity during weekdays and reduced business activity on weekends, especially Sundays, as visualized in the histogram.

Spatial Analysis

Spatial autocorrelation analysis using variograms revealed increasing semivariance with distance, supporting Tobler's first law, which states that closer observations tend to be more similar than distant ones.



The graph 1. The average rating spatial distribution in Pennsylvania

We fitted spatial linear models (`splm`) using the Matern covariance function, identifying significant explanatory variables influencing star ratings.

Spatial autocorrelation was confirmed, as ratings were more similar among closer locations. The fitted variogram models aligned with theoretical expectations, showing increasing semivariance with distance.

Specifically:

- **Credit card acceptance** significantly impacted star ratings in multiple states, notably IL, CA, IN, NJ, and DE.
- **Review counts** (popularity indicator) also significantly influenced ratings in several states, underscoring the relationship between popularity and consumer satisfaction.
- **Weekend operations** had a noticeable effect, particularly for businesses open on weekends, impacting ratings in select states.

Conclusion

Our analysis demonstrates that consumer star ratings exhibit notable spatial correlation. Variables such as acceptance of credit cards, review counts, and weekend operations significantly affect ratings and differ regionally. Future analyses could include additional business attributes or expand to broader geographical regions to provide deeper insights.

Appendix

Shiny Application

Author: Brian Cervantes Alvarez

Shiny Application

```r
# app.R

library(shiny)
library(shinyjs)
library(dplyr)
library(readr)
library(stringr)
library(sf)
library(leaflet)
library(plotly)
library(tidyr)
library(lubridate)
library(spmodel)
library(gstat)

#------------------------------
# 1) Load Final Data
#------------------------------
yelpDs <- read_csv("finalYelpData_sampled.csv") %>%
  filter(!is.na(latitude), !is.na(longitude))

#------------------------------
# 2) Helper Functions
#------------------------------
```

```r
convert_hours_str <- function(h) {
  if (is.na(h) || h == "Closed" || h == "00:00:00 - 00:00:00") {
    return("Closed")
  }
  parts <- strsplit(h, " - ")[[1]]
  if (length(parts) != 2) return("Closed")

  start_time <- strptime(parts[1], "%H:%M:%S")
  end_time   <- strptime(parts[2], "%H:%M:%S")

  start_str <- format(start_time, "%I:%M%p")
  end_str   <- format(end_time,   "%I:%M%p")

  # Remove leading zero from hour
  start_str <- sub("^0", "", start_str)
  end_str   <- sub("^0", "", end_str)

  paste(start_str, "-", end_str)
}

star_emoji <- Vectorize(function(rating) {
  if (is.na(rating) || rating <= 0) return("No rating")
  rating_rounded_half <- round(rating * 2) / 2
  if (rating_rounded_half > 5) rating_rounded_half <- 5
  full_stars <- floor(rating_rounded_half)
  half_star  <- (rating_rounded_half - full_stars) >= 0.5
  star_str   <- paste(rep(" ", full_stars), collapse="")
  if (half_star) star_str <- paste0(star_str, " ")
  paste0(rating, " (", star_str, ")")
})
```

```
modelMap <- c(Sph="Spherical", Exp="Exponential", Mat="Matern",
    Gau="Gaussian")

#-----------------------------
# 3) USER INTERFACE (UI)
#-----------------------------
ui <- withMathJax(fluidPage(
  useShinyjs(),
  tags$head(
    tags$link(

          href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@300;400;600&di
      rel="stylesheet"
    ),
    # Updated slider CSS: transparent unselected track, black
        active bar, orange handles
    tags$style(HTML("
      body {
        font-family: 'Open Sans', sans-serif;
        background-color: #FFFFFF;
        color: #000000;
        font-size: 2rem;
      }
      /* Tabset theme */
      .nav-tabs>li>a {
        color: #000000 !important; /* unselected tabs: black text
    */
      }
      .nav-tabs>li>a:hover {
```

```
    background-color: #D73F09 !important;
    color: #FFFFFF !important;
    border-radius: 8px 8px 0 0;
  }
  .nav-tabs>li.active>a,
  .nav-tabs>li.active>a:focus,
  .nav-tabs>li.active>a:hover {
    background-color: #D73F09 !important;
    color: #FFFFFF !important;
    border-radius: 8px 8px 0 0;
  }

  /* Slider theme:
     - transparent unselected track
     - black active bar
     - orange handles (#D73F09)
  */
  .js-range-slider .irs-line {
    background: transparent !important;
    border: none !important;
  }
  .js-range-slider .irs-bar,
  .js-range-slider .irs-bar-edge {
    background: #000000 !important; /* black active region */
    border-color: #000000 !important;
  }
  .js-range-slider .irs-handle>i,
  .js-range-slider .irs-handle>i:first-child,
  .js-range-slider .irs-handle>i:last-child {
    background: #D73F09 !important; /* orange handles */
  border-color: #D73F09 !important;
}
.js-range-slider .irs-single,
.js-range-slider .irs-to,
.js-range-slider .irs-from {
  background: #000000 !important;
  border-color: #000000 !important;
  color: #FFFFFF !important;
}

/* Box shadow class */
.boxShadow {
  box-shadow: rgba(0,0,0,0.25) 0px 54px 55px,
              rgba(0,0,0,0.12) 0px -12px 30px,
              rgba(0,0,0,0.12) 0px 4px 6px,
              rgba(0,0,0,0.17) 0px 12px 13px,
              rgba(0,0,0,0.09) 0px -3px 5px;
  border-radius: 15px;
}
#reviewMap {
  width: 100% !important;
  height: 75vh !important;
}
/* Top banner */
#topBanner {
  background-color: #D73F09;
  color: #FFFFFF;
  padding: 15px;
  margin: 10px;
  border-radius: 8px;
```

```
}
#yelpLogoContainer {
  text-align: right;
}
#yelpLogoContainer div {
  display: inline-block;
  background-color: #FFFFFF;
  padding: 5px;
  border-radius: 8px;
}
#yelpLogoContainer img {
  width: 60px;
}
.marker-cluster div {
  background-color: rgba(0,0,0,0.7) !important;
  border-radius: 20px;
  width: 40px;
  height: 40px;
  line-height: 40px;
  color: #FFFFFF !important;
  text-align: center;
}

/* Provide some vertical spacing for sidebars */
#map_sidebar,
#cc_sidebar,
#facet_sidebar,
#wh_sidebar,
#dist_sidebar,
#star_sidebar {
        margin-top: 15px;
      }
      .form-group {
        margin-bottom: 15px !important;
      }

      /* Black background, white text for selectInput */
      .selectize-control.single .selectize-input {
        background-color: #000000 !important;
        color: #FFFFFF !important;
        border: 1px solid #000000 !important;
      }
      .selectize-dropdown,
      .selectize-dropdown-content,
      .selectize-control.single .selectize-input.dropdown-active {
        background-color: #000000 !important;
        color: #FFFFFF !important;
        border: 1px solid #000000 !important;
      }
    "))
  ),

  fluidRow(
    id="topBanner",
    column(
      width=9,
      tags$h2("Yelp Reviews on Businesses Dashboard"),
      tags$p("This Shiny application showcases real-world data
    related to businesses and includes actual customer reviews.
    Explore star ratings, open hours, and spatial analysis with
    variogram modeling.")
    ),
    column(
      width=3,
      div(
        id="yelpLogoContainer",
        div(

              img(src="https://raw.githubusercontent.com/bcervantesalvarez/Portfolio/r
              alt="Yelp Logo")
        )
      )
    )
  ),
  br(),

  tabsetPanel(
    #=========================
    # TAB 1: SPATIAL MAP
    #=========================
    tabPanel("Spatial Map",
      fluidRow(
        column(
          width=3,
          id="map_sidebar",
          selectInput("citySelect", "City:", choices=c("All",
           sort(unique(yelpDs$city)))),
          sliderInput("ratingRange", "Star Rating:", min=1, max=5,
           value=c(1,5))
        ),
        column(
          width=9,
          id="map_main",
          actionButton("toggle_map", "Hide Filters"),
          tags$h4("About the Spatial Map"),
          tags$p("This interactive map displays the locations of
    Yelp businesses based on your selected city and star rating
    range. Individual markers represent each business, and when
    many businesses are located close together, they group into
    clusters that you can click to zoom in and view more detail."),
          div(class="boxShadow", leafletOutput("reviewMap",
           height="600px"))
        )
      )
    ),

    #==========================
    # TAB 2: DATA VISUALIZATION
    #==========================
    tabPanel("Data Visualization",
      tabsetPanel(
      # 2.1: Credit Cards vs. Star Rating
      tabPanel("Credit Cards vs. Star Rating",
        fluidRow(
          column(
            width=3,
            id="cc_sidebar",
           selectInput("ccCity", "City:", choices=c("All",
               sort(unique(yelpDs$city)))),
           sliderInput("ccStarRange", "Star Rating:", min=1,
               max=5, value=c(1,5))
            ),
            column(
           width=9,
           id="cc_main",
           actionButton("toggle_cc", "Hide Filters"),
           tags$h4("Credit Cards vs. Star Rating"),
           tags$p("This plot shows the proportion of businesses
    accepting credit cards across different star ratings. Filter by
    city and star rating range to narrow the results."),
           div(class="boxShadow",
               plotlyOutput("creditCardStarPlot", height="600px"))
            )
          )
          ),
        # 2.2: Open Hours by Day
        tabPanel("Open Hours by Day",
          fluidRow(
            column(
           width=3,
           id="facet_sidebar",
           selectInput("facetCity", "City:", choices=c("All",
               sort(unique(yelpDs$city)))),
           sliderInput("facetStarRange", "Star Rating:", min=1,
               max=5, value=c(1,5)),
           checkboxGroupInput("whichDays", "Days of Week to
               Show:",
           choices=c("Monday"="monday_open_hours",
                     "Tuesday"="tuesday_open_hours",
                     "Wednesday"="wednesday_open_hours",
                     "Thursday"="thursday_open_hours",
                     "Friday"="friday_open_hours",
                     "Saturday"="saturday_open_hours",
                     "Sunday"="sunday_open_hours"),
           selected=c("monday_open_hours",
                     "tuesday_open_hours",
                     "wednesday_open_hours",
                     "thursday_open_hours",
                     "friday_open_hours",
                     "saturday_open_hours",
                     "sunday_open_hours")
            )
          ),
          column(
            width=9,
            id="facet_main",
            actionButton("toggle_facet", "Hide Filters"),
            tags$h4("Open Hours by Day"),
            tags$p("Examine how businesses' daily open hours are
    distributed. Select which days to display, and filter by city
    and star rating."),
            div(class="boxShadow",
                plotlyOutput("openHoursDistribution",
                height="600px"))
          )
        )
        ),
      # 2.3: Weekly Hours vs. Star Rating
      tabPanel("Weekly Hours vs. Star Rating",
        fluidRow(
          column(
            width=3,
            id="wh_sidebar",
            selectInput("whCity", "City:", choices=c("All",
             sort(unique(yelpDs$city)))),
            sliderInput("whStarRange", "Star Rating:", min=1,
             max=5, value=c(1,5)),
            sliderInput("whHourRange", "Weekly Hours Range:",
             min=0, max=168, value=c(0,168))
          ),
          column(
            width=9,
            id="wh_main",
            actionButton("toggle_wh", "Hide Filters"),
            tags$h4("Weekly Hours vs. Star Rating"),
            tags$p("This bar chart shows average weekly open hours
    grouped by star rating. Adjust city, rating range, and weekly
    hours range to explore relationships."),
            div(class="boxShadow", plotlyOutput("openHoursVsStar",
             height="600px"))
          )
        )
        ),
      # 2.4: Distribution of Weekly Hours
      tabPanel("Distribution of Weekly Hours",
        fluidRow(
          column(
            width=3,
            id="dist_sidebar",
            selectInput("distCity", "City:", choices=c("All",
             sort(unique(yelpDs$city)))),
            sliderInput("distHourRange", "Weekly Hours Range:",
             min=0, max=168, value=c(0,168))
          ),
          column(
            width=9,
            id="dist_main",
            actionButton("toggle_dist", "Hide Filters"),
            tags$h4("Distribution of Weekly Hours"),
            tags$p("This histogram shows how many businesses fall
    into various weekly open-hour bins. Filter by city or choose a
    smaller hour range to focus on specific subsets."),
            div(class="boxShadow", plotlyOutput("weeklyHoursPlot",
             height="600px"))
          )
        )
        ),
      # 2.5: Distribution of Star Ratings
      tabPanel("Distribution of Star Ratings",
        fluidRow(
          column(
            width=3,
            id="star_sidebar",
            selectInput("starCity", "City:", choices=c("All",
             sort(unique(yelpDs$city)))),
            sliderInput("starFilterRange", "Star Rating Range:",
             min=1, max=5, value=c(1,5))
          ),
          column(
            width=9,
            id="star_main",
            actionButton("toggle_star", "Hide Filters"),
            tags$h4("Distribution of Star Ratings"),
            tags$p("View how star ratings are distributed among
    businesses. Filter by city or rating range to examine
    particular subsets."),
            div(class="boxShadow",
             plotlyOutput("distStarRatingPlot", height="600px"))
          )
        )
        )
      )
    ),

    #==========================
    # TAB 3: SPATIAL MODELING
    #==========================
    tabPanel("Spatial Modeling",
      sidebarLayout(
        sidebarPanel(
          tags$div(
            style="border:1px solid #ccc; background-color:#ffffcc;
             padding:8px; margin-bottom:10px;",
            "Warning: This model can take 15 seconds to 10 minutes
             depending on the State that is chosen."
          ),
          selectInput("stateChoice", "Choose a State:",
                      choices=sort(unique(yelpDs$state)),
                          selected="NJ"),
          selectInput("modelChoice", "Model Type:",
                      choices=c("Variogram only", "Variogram +
                          Fit")),
          selectInput("vgmType", "Variogram Model:",
                      choices=c("Sph", "Exp", "Mat", "Gau"),
                      selected="Sph"),
          numericInput("psillInit", "Initial Partial Sill:",
              value=0.1, min=0, max=999, step=0.01),
          numericInput("rangeInit", "Initial Range:", value=0.5,
              min=0, max=999, step=0.01),
          numericInput("nuggetInit", "Initial Nugget:", value=0.01,
              min=0, max=999, step=0.01),
          numericInput("kappaInit", "Kappa (Matern only):",
              value=1, min=0.01, max=5, step=0.01),
          actionButton("runModel", "Run Model")
        ),
        mainPanel(
          fluidRow(
            column(
              width=12,
              tags$h4("Spatial Modeling: Variogram Overview"),
              tags$p(
             "A variogram measures how the differences in your
                 data change with distance. We examine star
                 ratings to see potential spatial clustering.
                 One common formulation is ",
             withMathJax("$$\\gamma(h) = \\frac{1}{2N(h)}
                 \\sum_{i,j\\in h} [z(x_i)-z(x_j)]^2,$$"),
              " where \\(z(x_i)\\) is the logged star rating at
                  location \\(x_i\\) and \\(h\\) represents the
                  separation distance. Key parameters include:",
              tags$ul(
                tags$li(tags$strong("Partial Sill:"), " the
  variance contributed by spatial correlation."),
                tags$li(tags$strong("Range:"), " the distance at
  which autocorrelation becomes negligible."),
                tags$li(tags$strong("Nugget:"), "
  random/unexplained variance at zero distance."),
                tags$li(tags$strong("Kappa:"), " for the Matern
  model, controlling the shape of spatial decay.")
              ),
              "Adjust these parameters and choose a variogram
                  model (Spherical, Exponential, Gaussian, or
                  Matern) to see how star ratings might be
                  correlated over distance."
            )
          )
        ),
        verbatimTextOutput("modelSummary"),
        plotOutput("modelVisualization", width="700px",
            height="500px")
      )
    )
  )
 )
))

#-----------------------------
```

```
# 4) SERVER
#-------------------------------
server <- function(input, output, session) {

  #-------------------------------------
  # Show/hide sidebars from main panel
  #-------------------------------------
  hideSidebar <- function(sidebarID, mainID, buttonID) {
    runjs(sprintf('document.getElementById("%s").style.display =
        "none";', sidebarID))

        runjs(sprintf('$("#%s").removeClass("col-sm-9").addClass("col-sm-12");',
        mainID))
    updateActionButton(session, buttonID, label="Show Filters")
  }

  showSidebar <- function(sidebarID, mainID, buttonID) {
    runjs(sprintf('document.getElementById("%s").style.display =
        "block";', sidebarID))

        runjs(sprintf('$("#%s").removeClass("col-sm-12").addClass("col-sm-9");',
        mainID))
    updateActionButton(session, buttonID, label="Hide Filters")
  }

  # Track sidebar visibility
  mapVisible   <- reactiveVal(TRUE)
  ccVisible    <- reactiveVal(TRUE)
  facetVisible <- reactiveVal(TRUE)
  whVisible    <- reactiveVal(TRUE)
```

```r
distVisible  <- reactiveVal(TRUE)
starVisible  <- reactiveVal(TRUE)

observeEvent(input$toggle_map, {
  if(mapVisible()) {
    hideSidebar("map_sidebar", "map_main", "toggle_map")
    mapVisible(FALSE)
  } else {
    showSidebar("map_sidebar", "map_main", "toggle_map")
    mapVisible(TRUE)
  }
})
observeEvent(input$toggle_cc, {
  if(ccVisible()) {
    hideSidebar("cc_sidebar", "cc_main", "toggle_cc")
    ccVisible(FALSE)
  } else {
    showSidebar("cc_sidebar", "cc_main", "toggle_cc")
    ccVisible(TRUE)
  }
})
observeEvent(input$toggle_facet, {
  if(facetVisible()) {
    hideSidebar("facet_sidebar", "facet_main", "toggle_facet")
    facetVisible(FALSE)
  } else {
    showSidebar("facet_sidebar", "facet_main", "toggle_facet")
    facetVisible(TRUE)
  }
})
```

```r
observeEvent(input$toggle_wh, {
  if(whVisible()) {
    hideSidebar("wh_sidebar", "wh_main", "toggle_wh")
    whVisible(FALSE)
  } else {
    showSidebar("wh_sidebar", "wh_main", "toggle_wh")
    whVisible(TRUE)
  }
})
observeEvent(input$toggle_dist, {
  if(distVisible()) {
    hideSidebar("dist_sidebar", "dist_main", "toggle_dist")
    distVisible(FALSE)
  } else {
    showSidebar("dist_sidebar", "dist_main", "toggle_dist")
    distVisible(TRUE)
  }
})
observeEvent(input$toggle_star, {
  if(starVisible()) {
    hideSidebar("star_sidebar", "star_main", "toggle_star")
    starVisible(FALSE)
  } else {
    showSidebar("star_sidebar", "star_main", "toggle_star")
    starVisible(TRUE)
  }
})

#======================================================
# TAB 1: SPATIAL MAP
```

```
#=====================================================

filteredData <- reactive({
  data <- yelpDs
  if (input$citySelect != "All") {
    data <- data %>% filter(city == input$citySelect)
  }
  data
})

output$reviewMap <- renderLeaflet({
  leaflet() %>%
    addProviderTiles("CartoDB.Positron") %>%
    setView(
      lng = mean(yelpDs$longitude, na.rm=TRUE),
      lat = mean(yelpDs$latitude,  na.rm=TRUE),
      zoom=4
    )
})

observe({
  req(input$reviewMap_bounds, input$reviewMap_zoom)
  bounds <- input$reviewMap_bounds
  zoom   <- input$reviewMap_zoom

  data_in_bounds <- filteredData() %>%
    filter(latitude >= bounds$south, latitude <= bounds$north,
           longitude >= bounds$west,  longitude <= bounds$east)

  aggregated <- data_in_bounds %>%
    group_by(businessName, latitude, longitude) %>%
```

```r
summarize(
  starRatings    = list(starRating),
  reviewTexts    = list(text),
  total_reviews  = n(),
  avg_rating_num = mean(starRating, na.rm=TRUE),
  mondayH        = convert_hours_str(first(mondayHours)),
  tuesdayH       = convert_hours_str(first(tuesdayHours)),
  wednesdayH     = convert_hours_str(first(wednesdayHours)),
  thursdayH      = convert_hours_str(first(thursdayHours)),
  fridayH        = convert_hours_str(first(fridayHours)),
  saturdayH      = convert_hours_str(first(saturdayHours)),
  sundayH        = convert_hours_str(first(sundayHours)),
  .groups        = "drop"
) %>%
rowwise() %>%
mutate(
  idx_lowest  = which.min(starRatings),
  idx_highest = which.max(starRatings),
  idx_highest = if (idx_lowest == idx_highest &&
      length(starRatings) > 1) {
    if (idx_lowest == 1) 2 else 1
  } else idx_highest,
  lowest_star_num  = starRatings[idx_lowest],
  highest_star_num = starRatings[idx_highest],
  lowest_review    = reviewTexts[idx_lowest],
  highest_review   = reviewTexts[idx_highest]
) %>%
ungroup() %>%
filter(
  avg_rating_num >= input$ratingRange[1],
        avg_rating_num <= input$ratingRange[2]
      ) %>%
      mutate(
        avg_rating = star_emoji(round(avg_rating_num,1))
      )

    leafletProxy("reviewMap", data=aggregated) %>%
      clearMarkers() %>%
      clearPopups()

    if(nrow(aggregated)==0) return()

    # 3-part popup
    popup_html <- ~paste0(
      "<div style='width:300px;font-family:\"Open
          Sans\",sans-serif;box-shadow:0 4px 8px
          rgba(0,0,0,0.2);border-radius:5px;overflow:hidden;'>",
      "<div
          style='background-color:#000000;color:#FFFFFF;padding:10px;font-size:16px;f
      businessName,
      "</div>",
      "<div
          style='background-color:#FFFFFF;color:#000000;padding:10px;font-size:13px;'>
      "<strong>Hours:</strong><br>",
      "Mon: ", mondayH, "<br>",
      "Tue: ", tuesdayH, "<br>",
      "Wed: ", wednesdayH, "<br>",
      "Thu: ", thursdayH, "<br>",
      "Fri: ", fridayH, "<br>",
      "Sat: ", saturdayH, "<br>",
    "Sun: ", sundayH,
    "</div>",
    "<div
        style='background-color:#D73F09;color:#FFFFFF;padding:10px;font-size:13px;'>
    "<strong>Avg. Star Rating:</strong> ", avg_rating, "<br>",
    "<strong>Total Reviews:</strong> ", total_reviews, "<br>",
    "<hr style='margin:8px 0;border:none;border-top:1px solid
        #ccc;'>",
    "<strong>Lowest Review:</strong><br>",
    substring(lowest_review,1,100), "<br><br>",
    "<strong>Highest Review:</strong><br>",
    substring(highest_review,1,100),
    "</div>",
    "</div>"
  )

  if (zoom >= 16) {
    leafletProxy("reviewMap", data=aggregated) %>%
      addCircleMarkers(
        lng=~longitude, lat=~latitude,
        popup=popup_html,
        color="#000000",
        fillColor="#D73F09",
        fillOpacity=0.9,
        radius=6
      )
  } else {
    leafletProxy("reviewMap", data=aggregated) %>%
      addCircleMarkers(
        lng=~longitude, lat=~latitude,
            popup=popup_html,
            color="#000000",
            fillColor="#D73F09",
            fillOpacity=0.9,
            radius=6,
            clusterOptions=markerClusterOptions(
              iconCreateFunction=JS("
                function(cluster) {
                  var count = cluster.getChildCount();
                  return new L.DivIcon({
                    html: '<div title=\"This cluster shows the number
      of businesses in this area. Click to zoom in.\"
      style=\"background-color: rgba(0,0,0,0.7); border-radius: 20px;
      width: 40px; height: 40px; line-height: 40px; color: #FFFFFF;
      text-align: center;\">' + count + '</div>',
                    className: 'marker-cluster', iconSize: new
      L.Point(40, 40)
                  });
                }
              ")
            )
          )
      }
    })

    #======================================================
    # TAB 2: DATA VISUALIZATION
    #======================================================
    # 2.1: Credit Cards vs. Star Rating
    creditPlotData <- reactive({
```

```
  data <- yelpDs
  if (input$ccCity != "All") {
    data <- data %>% filter(city==input$ccCity)
  }
  data %>% filter(starRating>=input$ccStarRange[1],
                  starRating<=input$ccStarRange[2])
})
output$creditCardStarPlot <- renderPlotly({
  req(creditPlotData())
  df <- creditPlotData()
  p <- df %>%
    ggplot(aes(x=starRating, fill=acceptsCreditCards)) +

        geom_bar(aes(y=..count../tapply(..count..,..group..,sum)[..group..]),
             position="dodge") +
    scale_y_continuous(labels=scales::percent) +
    scale_x_continuous(breaks=seq(1,5,0.5)) +
    labs(title="Credit Cards vs. Star Rating",
         x="Star Rating",
         y="Percentage of Businesses",
         fill="Accepts CC?") +

        scale_fill_manual(values=c("FALSE"="#A7ACA2","TRUE"="#00859B"))
        +
    theme_minimal()
  ggplotly(p)
})

# 2.2: Open Hours by Day
facetPlotData <- reactive({
    data <- yelpDs
    if (input$facetCity != "All") {
      data <- data %>% filter(city==input$facetCity)
    }
    data <- data %>% filter(starRating>=input$facetStarRange[1],
                            starRating<=input$facetStarRange[2])
    if(length(input$whichDays)==0) return(data.frame())
    data
  })
  output$openHoursDistribution <- renderPlotly({
    req(facetPlotData())
    df <- facetPlotData()
    if(nrow(df)==0) {
      return(plotly_empty(type="scatter", title="No days selected
          or no data."))
    }
    dayCols <- input$whichDays
    if(!all(dayCols %in% names(df))) {
      return(plotly_empty(type="scatter", title="Chosen day columns
          not found."))
    }
    open_hours_long <- df %>% select(all_of(dayCols)) %>%
      pivot_longer(cols=everything(), names_to="day",
          values_to="open_hours")
    p <- open_hours_long %>%
      ggplot(aes(x=open_hours)) +
      geom_histogram(binwidth=1, fill="#006A8E") +
      facet_wrap(~ day, scales="free_y") +
      scale_x_continuous(breaks=seq(0,24,4)) +
      labs(title="Distribution of Open Hours by Selected Days",
          x="Open Hours", y="Count") +
      theme_minimal()
    ggplotly(p)
  })

  # 2.3: Weekly Hours vs. Star Rating
  whPlotData <- reactive({
    data <- yelpDs
    if (input$whCity!="All") {
      data <- data %>% filter(city==input$whCity)
    }
    data <- data %>%
      filter(starRating>=input$whStarRange[1],
             starRating<=input$whStarRange[2])
    if("weeklyHours" %in% names(data)) {
      data <- data %>% filter(weeklyHours>=input$whHourRange[1],
                               weeklyHours<=input$whHourRange[2])
    }
    data
  })
  output$openHoursVsStar <- renderPlotly({
    req(whPlotData())
    df <- whPlotData()
    p <- df %>%
      ggplot(aes(x=factor(starRating), y=weeklyHours)) +
      stat_summary(fun="mean", geom="bar", fill="#C4D6A4",
          color="black") +
      scale_x_discrete(breaks=seq(1,5,0.5)) +
      labs(title="Weekly Hours vs. Star Rating",
          x="Star Rating", y="Avg Weekly Hours") +
      theme_minimal()
    ggplotly(p)
  })

  # 2.4: Distribution of Weekly Hours
  distPlotData <- reactive({
    data <- yelpDs
    if (input$distCity!="All") {
      data <- data %>% filter(city==input$distCity)
    }
    if("weeklyHours" %in% names(data)) {
      data <- data %>% filter(weeklyHours>=input$distHourRange[1],
                               weeklyHours<=input$distHourRange[2])
    }
    data
  })
  output$weeklyHoursPlot <- renderPlotly({
    req(distPlotData())
    df <- distPlotData()
    p <- df %>%
      ggplot(aes(x=weeklyHours)) +
      geom_histogram(binwidth=5, fill="#FDD26E", color="black") +
      labs(title="Distribution of Weekly Hours",
          x="Weekly Hours", y="Count") +
      theme_minimal()
    ggplotly(p)
  })

  # 2.5: Distribution of Star Ratings
  starDistData <- reactive({
    data <- yelpDs
    if (input$starCity!="All") {
      data <- data %>% filter(city==input$starCity)
    }
    data %>% filter(starRating>=input$starFilterRange[1],
                    starRating<=input$starFilterRange[2])
  })
  output$distStarRatingPlot <- renderPlotly({
    req(starDistData())
    df <- starDistData()
    p <- df %>%
      ggplot(aes(x=starRating)) +
      geom_histogram(binwidth=0.5, fill="#C6DAE7", color="black") +
      scale_x_continuous(breaks=seq(1,5,0.5)) +
      labs(title="Distribution of Star Ratings",
          x="Star Rating", y="Count") +
      theme_minimal()
    ggplotly(p)
  })

  #======================================================
  # TAB 3: SPATIAL MODELING
  #======================================================
  modelData <- reactiveValues(
    varioObj=NULL,
    fittedVario=NULL,
    message="No model run yet."
  )

  observeEvent(input$runModel, {
```

```r
# Clear old results each time user re-runs
modelData$varioObj    <- NULL
modelData$fittedVario <- NULL
modelData$message     <- "No model run yet."

showNotification("Running spatial model... Please wait.",
    type="message", duration=NULL)
withProgress(message="Computing model...", value=0, {
  incProgress(0.2, detail="Filtering data by state...")
  state_df <- yelpDs %>% filter(state==input$stateChoice)
  if(nrow(state_df)<10) {
    modelData$message <- paste("Not enough data for state:",
input$stateChoice)
    return()
  }
  incProgress(0.4, detail="Converting data to spatial
      format...")
  state_sf <- st_as_sf(state_df,
coords=c("longitude","latitude"), crs=4326)
  state_sp <- as(state_sf, "Spatial")

  incProgress(0.6, detail="Filtering out invalid star
      ratings...")
  starPos <- state_sp@data$starRating>0
  sp_for_vario <- state_sp[starPos,]
  if(nrow(sp_for_vario@data)<10) {
    modelData$message <- paste("Not enough valid starRating for
state:", input$stateChoice)
    return()
  }
  incProgress(0.8, detail="Computing variogram...")
  vario <- variogram(log(starRating)~1, data=sp_for_vario)
  modelData$varioObj <- vario
  modelData$message <- paste("Variogram computed for",
input$stateChoice)

  if(tolower(input$modelChoice)=="variogram + fit") {
    incProgress(0.9, detail=paste("Fitting",
        modelMap[[input$vgmType]], "variogram model..."))
    init_psill  <- input$psillInit
    init_range  <- input$rangeInit
    init_nugget <- input$nuggetInit
    init_kappa  <- if(input$vgmType=="Mat") input$kappaInit
else 0

    init_vgm <- vgm(
      model  = input$vgmType,
      psill  = init_psill,
      range  = init_range,
      nugget = init_nugget,
      kappa  = init_kappa
    )

    fit_v <- tryCatch({
      fit.variogram(vario, model=init_vgm)
    }, error=function(e){ NULL })

    if(is.null(fit_v)) {
      modelData$fittedVario <- NULL
```

```
          modelData$message <- paste("Error fitting",
    modelMap[[input$vgmType]],
                                       "model for",
                                           input$stateChoice)
        } else {
          modelData$fittedVario <- fit_v
          modelData$message <- paste("Variogram +",
    modelMap[[input$vgmType]],
                                       "fit done for",
                                           input$stateChoice)
        }
      }
    })
    showNotification("Model run complete!", type="message",
        duration=5)
  })

  output$modelSummary <- renderPrint({
    if(is.null(modelData$varioObj)) {
      cat(modelData$message)
    } else {
      cat(modelData$message, "\n\n")
      cat("Sample Variogram:\n")
      print(modelData$varioObj)
      if(!is.null(modelData$fittedVario)) {
        cat("\nFitted Variogram:\n")
        print(modelData$fittedVario)
      }
    }
  })
  output$modelVisualization <- renderPlot({
    if(is.null(modelData$varioObj)) {
      plot(1:10,1:10,main="No variogram computed yet.")
      return()
    }
    vario <- modelData$varioObj
    if(is.null(modelData$fittedVario)) {
      plot(vario, main=paste("Sample Variogram -",
          input$stateChoice))
    } else {
      plot(vario, modelData$fittedVario,
           main=paste0("Variogram + ", modelMap[[input$vgmType]], "
               Fit - ", input$stateChoice))
    }
  })
}

shinyApp(ui, server)
```

Data Visualization

Author: Casey Nagai

```
---
title: "Final Project Code"
format: html
editor: visual
---
```

```r
library(ggplot2)
library(sf)
library(mapview)
library(lubridate)
library(stringr)
library(dplyr)
library(tidyr)

yelp_data = read.csv("yelpBusiness.csv")
```

Our dataset contains information about which businesses accept payment by credit card. We can see some noticeable differences between the distributions of star ratings among businesses which accept credit card and those who don't. Since there are many more of the former, we have normalized the counts by the number of businesses of each type.

```r
ggplot(yelp_data, mapping=aes(x=starRating,
    fill=acceptsCreditCards)) +
  geom_bar(aes(y=after_stat(c(
    count[group==1] / sum(count[group==1]),
    count[group==2] / sum(count[group==2])
  ))), position="dodge") +
  xlab("Star Rating") +
  scale_x_continuous(breaks=c(1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5)) +
  ylab("Percentage of Businesses") +
  scale_y_continuous(labels = scales::percent) +
  scale_fill_discrete(name = "Accepts Credit Cards", labels =
      c("Does not accept", "Accepts"), guide =
      guide_legend(reverse=TRUE))
```

```
convert_hours <- function(hour_str) {
  # If closed or NA, return 0 hours
  if(is.na(hour_str) || hour_str == "Closed") return(0)

  # Split string into start and end times
  parts <- str_split(hour_str, " - ", simplify = TRUE)
  start_time <- lubridate::hms(parts[1])
  end_time   <- lubridate::hms(parts[2])

  # Compute duration in hours
  duration <- as.numeric(end_time - start_time, units = "hours")
  if(duration <= 0) duration <- duration + 24  # Adjust for
      overnight shifts

  return(duration)
}

yelp_data$monday_open_hours <- sapply(yelp_data$mondayHours,
    convert_hours)
yelp_data$tuesday_open_hours <- sapply(yelp_data$tuesdayHours,
    convert_hours)
```

```
yelp_data$wednesday_open_hours <- sapply(yelp_data$wednesdayHours,
    convert_hours)
yelp_data$thursday_open_hours <- sapply(yelp_data$thursdayHours,
    convert_hours)
yelp_data$friday_open_hours <- sapply(yelp_data$fridayHours,
    convert_hours)
yelp_data$saturday_open_hours <- sapply(yelp_data$saturdayHours,
    convert_hours)
yelp_data$sunday_open_hours <- sapply(yelp_data$sundayHours,
    convert_hours)

yelp_data$weeklyHours = yelp_data[c("monday_open_hours",
    "tuesday_open_hours", "wednesday_open_hours",
    "thursday_open_hours", "friday_open_hours",
    "saturday_open_hours", "sunday_open_hours")] %>%
  rowSums()

open_hours_long <- yelp_data %>%
  select(monday_open_hours, tuesday_open_hours,
      wednesday_open_hours,
         thursday_open_hours, friday_open_hours,
    saturday_open_hours, sunday_open_hours) %>%
  pivot_longer(cols = everything(), names_to = "day", values_to =
      "open_hours")
```

We also have data on the open hours of businesses, so we can look at the distribution of open hours across the week. It seems that activity is lowest on the weekends, especially Sunday, which makes sense.

```
ggplot(open_hours_long, aes(x = open_hours)) +
  geom_histogram(binwidth = 1, fill = "steelblue") +
  facet_wrap(~ day) +
  scale_x_continuous(breaks = seq(0, 24, by = 4)) +
  labs(title = "Distribution of Open Hours by Day", x = "Open
      Hours", y = "Count")
```

Interestingly enough, there doesn't seem to be a strong relationship between open hours and popularity. The spike at 1.5 stars is also noteworthy.

```
ggplot(yelp_data, mapping=aes(x=starRating, y=weeklyHours)) +
  geom_bar(stat="summary", fun = "mean", fill="steelblue",
      colour="black") +
  scale_x_continuous(breaks=c(1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5)) +
  xlab("Star Rating") +
  ylab("Average Open Hours per Week")
```

Finally, we can look at the geographical distribution of stores and their hours on a map. In many places, we can see clusters of 24-hour stores located close together.

```
yelp_data_sf = st_as_sf(yelp_data, coords=c("longitude",
    "latitude"), crs=4326)
mapview(yelp_data_sf, zcol="weeklyHours")
```

Spatial Analysis

Author: Polina Iasakova

```
---

title: "spatial_project"

format: pdf

editor: visual

---
```

```r
library(tidyverse)

library(readr)

library(sp)

library(sf)

library(mapview)

library(gstat)

library(spmodel)


library(spData) # Please get version 2.3.3, not 2.3.4.

library(spdep)

library(versions)
```

```r
# install.versions('spData', '2.3.3')

#

# require(remotes)

# install_version("spData", version = "2.3.3")
```

```r
yelpBusinessMerged <- read.csv("yelpBusinessReviewsMerged.csv")
```

```
yelpBusinessMerged$acceptsCreditCards[yelpBusinessMerged$acceptsCreditCards==TRUE]
    = 1
yelpBusinessMerged$acceptsCreditCards[yelpBusinessMerged$acceptsCreditCards==FALSE]
    = 0

yelpBusinessMerged$reviewCount_perc =
    yelpBusinessMerged$reviewCount /
    max(yelpBusinessMerged$reviewCount)

yelpBusinessMerged$weekends = (yelpBusinessMerged$saturdayHours !=
    'Closed') + (yelpBusinessMerged$sundayHours != 'Closed')
```

```
unique(yelpBusinessMerged$state)


paste('PA' , sum(yelpBusinessMerged$state == 'PA'))
paste('FL' , sum(yelpBusinessMerged$state == 'FL'))
paste('LA', sum(yelpBusinessMerged$state == 'LA'))
paste('TN' , sum(yelpBusinessMerged$state == 'TN'))
paste('NV' , sum(yelpBusinessMerged$state == 'NV'))
paste('MO' , sum(yelpBusinessMerged$state == 'MO'))
paste('IN' , sum(yelpBusinessMerged$state == 'IN'))
paste('CA' , sum(yelpBusinessMerged$state == 'CA'))
paste('AZ' , sum(yelpBusinessMerged$state == 'AZ'))
paste('NJ' , sum(yelpBusinessMerged$state == 'NJ'))
paste('ID' , sum(yelpBusinessMerged$state == 'ID'))
paste('DE' , sum(yelpBusinessMerged$state == 'DE'))
paste('IL' , sum(yelpBusinessMerged$state == 'IL'))
paste('SD' , sum(yelpBusinessMerged$state == 'SD'))
paste('WA' , sum(yelpBusinessMerged$state == 'WA'))
```

```
df_LA = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'LA'),]
yelp_la <- st_as_sf(df_LA, coords = c("longitude", "latitude"), crs
    = 28992)

df_PA = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'PA'),]
yelp_pa <- st_as_sf(df_PA, coords = c("longitude", "latitude"), crs
    = 28992)

df_IN = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'IN'),]
yelp_in <- st_as_sf(df_IN, coords = c("longitude", "latitude"), crs
    = 28992)

df_FL = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'FL'),]
yelp_fl <- st_as_sf(df_FL, coords = c("longitude", "latitude"), crs
    = 28992)

df_TN = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'TN'),]
yelp_tn <- st_as_sf(df_TN, coords = c("longitude", "latitude"), crs
    = 28992)

df_NV = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'NV'),]
yelp_nv <- st_as_sf(df_NV, coords = c("longitude", "latitude"), crs
    = 28992)
```

```
df_AZ = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'AZ'),]
yelp_az <- st_as_sf(df_AZ, coords = c("longitude", "latitude"), crs
    = 28992)

df_DE = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'DE'),]
yelp_de <- st_as_sf(df_DE, coords = c("longitude", "latitude"), crs
    = 28992)

df_MO = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'MO'),]
yelp_mo <- st_as_sf(df_MO, coords = c("longitude", "latitude"), crs
    = 28992)

df_ID = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'ID'),]
yelp_id <- st_as_sf(df_ID, coords = c("longitude", "latitude"), crs
    = 28992)

df_NJ = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'NJ'),]
yelp_nj <- st_as_sf(df_NJ, coords = c("longitude", "latitude"), crs
    = 28992)

df_CA = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'CA'),]
yelp_ca <- st_as_sf(df_CA, coords = c("longitude", "latitude"), crs
    = 28992)
```

```
df_IL = yelpBusinessMerged[which(yelpBusinessMerged$state ==
    'IL'),]
yelp_il <- st_as_sf(df_IL, coords = c("longitude", "latitude"), crs
    = 28992)
```

```
ggplot(data=yelp_la,) + geom_sf(aes(color=starRating)) +
    ggtitle('LA')
ggplot(data=yelp_in) + geom_sf(aes(color=starRating)) +
    ggtitle('IN')
ggplot(data=yelp_fl) + geom_sf(aes(color=starRating)) +
    ggtitle('FL')
ggplot(data=yelp_tn) + geom_sf(aes(color=starRating)) +
    ggtitle('TN')
ggplot(data=yelp_nv) + geom_sf(aes(color=starRating)) +
    ggtitle('NV')
ggplot(data=yelp_az) + geom_sf(aes(color=starRating)) +
    ggtitle('AZ')
ggplot(data=yelp_de) + geom_sf(aes(color=starRating)) +
    ggtitle('DE')
ggplot(data=yelp_mo) + geom_sf(aes(color=starRating)) +
    ggtitle('MO')
ggplot(data=yelp_id) + geom_sf(aes(color=starRating)) +
    ggtitle('ID')
ggplot(data=yelp_nj) + geom_sf(aes(color=starRating)) +
    ggtitle('NJ')
ggplot(data=yelp_ca) + geom_sf(aes(color=starRating)) +
    ggtitle('CA')
ggplot(data=yelp_il) + geom_sf(aes(color=starRating)) +
    ggtitle('IL')
```

```r
yelp_PA <- st_as_sf(df_PA, coords = c("longitude", "latitude"), crs
    = 28992)
```

```r
v <- variogram(log(starRating) ~ 1, yelp_PA)
plot(v)
```

```r
v_fit_PA <- fit.variogram(object=v,
                          model=vgm(psill = 0.1, model = "Mat",
                                    range = 0.4, nugget = 0.04,
                                        kappa = 1))
v_fit_PA

# Plot the fitted spherical variogram with the sample variogram.
plot(v, v_fit_PA)
```

```r
# yelp_gstat <- gstat(formula=starRating ~ reviewCount + cool +
    funny + useful + acceptsCreditCards, data=yelp_sf,
    model=v_sph_fit)
# yelp_gstat
# # predict(yelp_gstat, newdata=yelp_pred)
```

```r
start_time <- Sys.time()



yelp_splm_PA <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_PA,
    spcov_type='matern')
summary(yelp_splm_PA)
```

```
end_time <- Sys.time()
end_time - start_time
```

```
save(yelp_splm_PA, file="yelp_splm_PA.Rdata")
```

```
# ggplot(data=yelp_sf) + geom_sf(aes(color=starRating))
#                                # size=acceptsCreditCards))
# ggplot(data=yelp_sf) + geom_point(aes(x=reviewCount,
    y=starRating), size=3)
```

FL 70732

```
yelp_FL <- st_as_sf(df_FL, coords = c("longitude", "latitude"), crs
    = 28992)
```

```
v <- variogram(log(starRating) ~ 1, yelp_FL)
plot(v)
```

```
v_fit_FL <- fit.variogram(object=v,
                             model=vgm(psill = 0.1, model = "Mat",
                                       range = 0.4, nugget = 0.04,
                                           kappa = 1))
v_fit_FL

# Plot the fitted spherical variogram with the sample variogram.
plot(v, v_fit_FL)
```

```
start_time <- Sys.time()
```

```r
yelp_splm_FL <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_FL,
    spcov_type='matern')
summary(yelp_splm_FL)

end_time <- Sys.time()
end_time - start_time
```

```r
save(yelp_splm_FL, file="yelp_splm_FL.Rdata")
```

LA 52027

```r
yelp_LA <- st_as_sf(df_LA, coords = c("longitude", "latitude"), crs
    = 28992)
```

```r
v <- variogram(log(starRating) ~ 1, yelp_LA)
plot(v)
```

```r
v_fit_LA <- fit.variogram(object=v,
                             model=vgm(psill = 0.1, model = "Mat",
                                        range = 0.4, nugget = 0.04,
                                           kappa = 1))
v_fit_LA

# Plot the fitted spherical variogram with the sample variogram.
plot(v, v_fit_LA)
```

```r
start_time <- Sys.time()



yelp_splm_LA <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_LA,
    spcov_type='matern')
summary(yelp_splm_LA)

end_time <- Sys.time()
end_time - start_time
```

```r
save(yelp_splm_LA, file="yelp_splm_LA.Rdata")
```

TN 44232

```r
yelp_TN <- st_as_sf(df_TN, coords = c("longitude", "latitude"), crs
    = 28992)
```

```r
v <- variogram(log(starRating) ~ 1, yelp_TN)
plot(v)
```

```r
v_fit_TN <- fit.variogram(object=v,
                          model=vgm(psill = 0.1, model = "Mat",
                                    range = 0.4, nugget = 0.04,
                                        kappa = 1))
v_fit_TN

# Plot the fitted spherical variogram with the sample variogram.
plot(v, v_fit_TN)
```

```r
start_time <- Sys.time()



yelp_splm_TN <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_TN,
    spcov_type='matern')
summary(yelp_splm_TN)

end_time <- Sys.time()
end_time - start_time
```

```r
save(yelp_splm_TN, file="yelp_splm_TN.Rdata")
```

NV 34655

```r
yelp_NV <- st_as_sf(df_NV, coords = c("longitude", "latitude"), crs
    = 28992)
```

```r
v <- variogram(log(starRating) ~ 1, yelp_NV)
plot(v)
```

```r
v_fit_NV <- fit.variogram(object=v,
                          model=vgm(psill = 0.1, model = "Mat",
                                    range = 0.4, nugget = 0.06,
                                        kappa = ))
v_fit_NV

# Plot the fitted spherical variogram with the sample variogram.
plot(v, v_fit_NV)
```

```r
start_time <- Sys.time()



yelp_splm_NV <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_NV,
    spcov_type='matern')
summary(yelp_splm_NV)

end_time <- Sys.time()
end_time - start_time
```

```r
save(yelp_splm_NV, file="yelp_splm_NV.Rdata")
```

MO 32717

```r
yelp_MO <- st_as_sf(df_MO, coords = c("longitude", "latitude"), crs
    = 28992)
```

```r
v <- variogram(log(starRating) ~ 1, yelp_MO)
plot(v)
```

```r
v_fit_MO <- fit.variogram(object=v,
                             model=vgm(psill = 50, model = "Hol",
                                       range = 0.7, nugget = 0.059))
v_fit_MO

# Plot the fitted spherical variogram with the sample variogram.
plot(v, v_fit_MO)
```

```r
start_time <- Sys.time()



yelp_splm_MO <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_MO,
    spcov_type='matern')
summary(yelp_splm_MO)

end_time <- Sys.time()
end_time - start_time
```

```r
save(yelp_splm_MO, file="yelp_splm_MO.Rdata")
```

IN 29343

```r
yelp_IN <- st_as_sf(df_IN, coords = c("longitude", "latitude"), crs
    = 28992)
```

```r
v <- variogram(log(starRating) ~ 1, yelp_IN)
plot(v)
```

```r
v_fit_IN <- fit.variogram(object=v,
                          model=vgm(psill = 0.1, model = "Mat",
                                    range = 0.4, nugget = 0.04,
                                        kappa = 1))
v_fit_IN

# Plot the fitted spherical variogram with the sample variogram.
plot(v, v_fit_IN)
```

```
start_time <- Sys.time()



yelp_splm_IN <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_IN,
    spcov_type='matern')
summary(yelp_splm_IN)

end_time <- Sys.time()
end_time - start_time
```

```
save(yelp_splm_IN, file="yelp_splm_IN.Rdata")
```

CA 29281

```
yelp_CA <- st_as_sf(df_CA, coords = c("longitude", "latitude"), crs
    = 28992)
```

```
v <- variogram(log(starRating) ~ 1, yelp_CA)
plot(v)
```

```
v_fit_CA <- fit.variogram(object=v,
                          model=vgm(psill = 0.1, model = "Mat",
                                    range = 0.4, nugget = 0.04,
                                        kappa = 1))
v_fit_CA

# Plot the fitted spherical variogram with the sample variogram.
plot(v, v_fit_CA)
```

```r
start_time <- Sys.time()



yelp_splm_CA <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_CA,
    spcov_type='matern')
summary(yelp_splm_CA)

end_time <- Sys.time()
end_time - start_time
```

```r
save(yelp_splm_CA, file="yelp_splm_CA.Rdata")
```

AZ 27397

```r
yelp_AZ <- st_as_sf(df_AZ, coords = c("longitude", "latitude"), crs
    = 28992)
```

```r
v <- variogram(log(starRating) ~ 1, yelp_AZ)
plot(v)
```

```r
v_fit_AZ <- fit.variogram(object=v,
                          model=vgm(psill = 0.1, model = "Mat",
                                    range = 0.4, nugget = 0.06,
                                        kappa = 1))
v_fit_AZ

# Plot the fitted spherical variogram with the sample variogram.
plot(v, v_fit_AZ)
```

```
#AZ

start_time <- Sys.time()

yelp_splm_AZ <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_AZ,
    spcov_type='matern')
summary(yelp_splm_AZ)

end_time <- Sys.time()
end_time - start_time
```

```
save(yelp_splm_AZ, file="yelp_splm_AZ.Rdata")
```

NJ 14624

```
yelp_NJ <- st_as_sf(df_NJ, coords = c("longitude", "latitude"), crs
    = 28992)
```

```
v <- variogram(log(starRating) ~ 1, yelp_NJ)
plot(v)
```

```
v_fit_NJ <- fit.variogram(object=v,
                          model=vgm(psill = 0.1, model = "Mat",
                                    range = 0.4, nugget = 0.04,
                                        kappa = 1))
v_fit_NJ

# Plot the fitted spherical variogram with the sample variogram.
plot(v, v_fit_NJ)
```

```
start_time <- Sys.time()



yelp_splm_NJ <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_NJ,
    spcov_type='matern')
summary(yelp_splm_NJ)

end_time <- Sys.time()
end_time - start_time
```

```
save(yelp_splm_NJ, file="yelp_splm_NJ.Rdata")
```

ID 10218

```
yelp_ID <- st_as_sf(df_ID, coords = c("longitude", "latitude"), crs
    = 28992)
```

```
v <- variogram(log(starRating) ~ 1, yelp_ID)
plot(v)
```

```
v_fit_ID <- fit.variogram(object=v,
                          model=vgm(psill = 0.1, model = "Mat",
                                    range = 0.004, nugget = 0.04,
                                        kappa = 1))
v_fit_ID

# Plot the fitted spherical variogram with the sample variogram.
plot(v, v_fit_ID)
```

```r
#ID

start_time <- Sys.time()


yelp_splm_ID <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_ID,
    spcov_type='matern')
summary(yelp_splm_ID)

end_time <- Sys.time()
end_time - start_time
```

```r
save(yelp_splm_ID, file="yelp_splm_ID.Rdata")
```

DE 5630

```r
yelp_DE <- st_as_sf(df_DE, coords = c("longitude", "latitude"), crs
    = 28992)
```

```r
v <- variogram(log(starRating) ~ 1, yelp_DE)
plot(v)
```

```r
v_fit_DE <- fit.variogram(object=v,
                          model=vgm(psill = 0.8, model = "Mat",
                                    range = 0.4, nugget = 0.06,
                                        kappa = 1))
v_fit_DE

# Plot the fitted spherical variogram with the sample variogram.
```

```r
plot(v, v_fit_DE)
```

```r
start_time <- Sys.time()



yelp_splm_DE <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_DE,
    spcov_type='matern')
summary(yelp_splm_DE)

end_time <- Sys.time()
end_time - start_time
```

```r
save(yelp_splm_DE, file="yelp_splm_DE.Rdata")
```

IL 3451

```r
yelp_IL <- st_as_sf(df_IL, coords = c("longitude", "latitude"), crs
    = 28992)
```

```r
v <- variogram(log(starRating) ~ 1, yelp_IL)
plot(v)
```

```r
v_fit_IL <- fit.variogram(object=v,
                            model=vgm(psill = 0.8, model = "Mat",
                                      range = 0.003, nugget = 0.06,
                                        kappa = 1))
v_fit_IL

```

```r
# Plot the fitted spherical variogram with the sample variogram.
plot(v, v_fit_IL)
```

```r
start_time <- Sys.time()



yelp_splm_IL <- splm(starRating ~ reviewCount + cool + funny +
    useful + acceptsCreditCards + weekends, data=yelp_IL,
    spcov_type='matern')
summary(yelp_splm_IL)

end_time <- Sys.time()
end_time - start_time
```

```r
save(yelp_splm_IL, file="yelp_splm_IL.Rdata")
```