As a data analyst you want to provide clear cut insights for your end users, enabling them to extract all the business value provided by your solution. If your end user is data and analytical savvy then explaining results might be a piece of cake. Unfortunately not all stakeholders are able to fully grab the potential of your solution. In that scenario data visualisation is a powerful concept. Data visualisation enables you to provide the core insights in a single graphic. In this video we demonstrate how to easily visualize data with a geographical component in R using leaflet.  
*(the code can be found below the video)*

In this example we use open source government data: data on characteristics of [inhabitants](https://utrecht.buurtmonitor.nl/) and a [GeoJSON datafile](https://www.dataplatform.nl/resource/ddc3fcba-ba33-47d4-8ef8-fad31b4013bc). GeoJSON is a popular data format for many geographical technologies and services mainly because it’s simple, lightweight and straightforward. We use the leaflet package in R because it handles GeoJSON data very well straight out of the box.

# packages required

library(geojsonio)

library(leaflet)

library(magrittr)

library(htmlwidgets)

#######################

## READ POLYGON DATA ##

#######################

# Read shapefile: Spatial Polygon DB

neighborhoods\_utrecht <- geojson\_read("neighborhoods-utrecht.geojson", method= "local", what = "sp")

# What kind of data does this spatial object contain

head(neighborhoods\_utrecht@data)

# Show the neighborhoods of Utrecht on a Leaflet map

leaflet(neighborhoods\_utrecht) %>%

#addProviderTiles("nlmaps.standaard") %>%

addProviderTiles("Esri.WorldGrayCanvas") %>%

addPolygons(stroke = TRUE, color = "white", weight="1", smoothFactor = 0.3, fillOpacity = 0.7, fillColor = "lightblue")

##################################

## LOAD DATA TO PLOT ON POLYGON ##

##################################

# Import data to be displayed on the map

utrecht\_data <- read.csv("data\_utrecht.csv", header = TRUE, sep = ";", quote = "\"", dec = ".", fill = TRUE)

# Create merge ID and merge data

neighborhoods\_utrecht@data$gwb\_buurt\_code <- 344 \* 10000 + as.numeric(levels(neighborhoods\_utrecht@data$KODE))[neighborhoods\_utrecht@data$KODE]

neighborhoods\_utrecht@data <- merge(neighborhoods\_utrecht@data, utrecht\_data, by.x="gwb\_buurt\_code", by.y="gwb\_buurt\_code")

#########################

## DISPLAY DATA ON MAP ##

#########################

# Define cut points for the colorbins

cuts <- c(0.0, 0.05, 0.1, 0.15, 0.20, 0.25, 0.30, 1)

# Choose a color palette and assign it to the values

colorbins <- colorBin("YlOrRd", domain = neighborhoods\_utrecht$p\_65\_inf, bins = cuts)

# Display data on elderly people on the map

map <- leaflet(neighborhoods\_utrecht) %>%

addTiles() %>%

addProviderTiles("Esri.WorldGrayCanvas") %>%

addPolygons(stroke = TRUE, color = "white", weight="1", smoothFactor = 0.3,

fillOpacity = 0.7, fillColor = ~colorbins(neighborhoods\_utrecht$p\_65\_inf))

map

# Add a legend

map\_with\_legend <- map %>% addLegend(pal = colorbins,

values = neighborhoods\_utrecht$p\_65\_inf,

labFormat = labelFormat(suffix = " %", transform = function(p\_65\_inf) 100 \* p\_65\_inf),

opacity = 0.7, title = "Residents of age 65 and older", position = "topright")

map\_with\_legend

###########################################

## ADD MOUSE-OVER HIGHLIGHTS AND TOOLTIP ##

###########################################

# Create HTML labels for tooltip

tooltip <- sprintf("**%s**  
%.1f%% elderly residents"

,neighborhoods\_utrecht$gwb\_buurt\_naam

,neighborhoods\_utrecht$p\_65\_inf\*100

) %>% lapply(htmltools::HTML)

# Display map

map\_with\_tooltip <- map\_with\_legend %>% addPolygons(stroke = TRUE, color = "white", weight="1", smoothFactor = 0.3, fillOpacity = 0.7,

fillColor = ~colorbins(neighborhoods\_utrecht$p\_65\_inf),

highlight = highlightOptions(weight = 5, color = "grey", fillOpacity = 0.7, bringToFront = TRUE),

label = tooltip

)

map\_with\_tooltip

###########################

## SAVE OUTPUT HTML FILE ##

###########################

# Save output as HTML widget (or incorporate into Shiny / Flexdashboard)

saveWidget(map\_with\_tooltip, file="Elderly residents in Utrecht.html")

The end result is a portable HTML file you can use in reports, on a webpage or in a R dashboard (flexdashboard / shiny). This video covers the basics of visualising geographical data in R with leaflet. There are many options that can be added quite easily.

A short note on data security: the output of the leaflet package is a standalone html-file, so no information is send to a web server. To be completely safe, only add data to the GeoJSON data matrix that you are willing to share with the public you provide access to the output file.

**Examples**

x <- c(1:9, 8:1)

y <- c(1, 2\*(5:3), 2, -1, 17, 9, 8, 2:9)

op <- par(mfcol = c(3, 1))

for(xpd in c(FALSE, TRUE, NA)) {

plot(1:10, main = paste("xpd =", xpd))

box("figure", col = "pink", lwd = 3)

polygon(x, y, xpd = xpd, col = "orange", lty = 2, lwd = 2, border = "red")

}

par(op)

n <- 100

xx <- c(0:n, n:0)

yy <- c(c(0, cumsum(stats::rnorm(n))), rev(c(0, cumsum(stats::rnorm(n)))))

plot (xx, yy, type = "n", xlab = "Time", ylab = "Distance")

polygon(xx, yy, col = "gray", border = "red")

title("Distance Between Brownian Motions")

# Multiple polygons from NA values

# and recycling of col, border, and lty

op <- par(mfrow = c(2, 1))

plot(c(1, 9), 1:2, type = "n")

polygon(1:9, c(2,1,2,1,1,2,1,2,1),

col = c("red", "blue"),

border = c("green", "yellow"),

lwd = 3, lty = c("dashed", "solid"))

plot(c(1, 9), 1:2, type = "n")

polygon(1:9, c(2,1,2,1,NA,2,1,2,1),

col = c("red", "blue"),

border = c("green", "yellow"),

lwd = 3, lty = c("dashed", "solid"))

par(op)

# Line-shaded polygons

plot(c(1, 9), 1:2, type = "n")

polygon(1:9, c(2,1,2,1,NA,2,1,2,1),

density = c(10, 20), angle = c(-45, 45))