Spatial Economics – Assignment 1

Max Heinze (h11742049@s.wu.ac.at) Kevin Kain (h12232066@s.wu.ac.at)
Jaime Miravet (h12235992@s.wu.ac.at)

April 2, 2024

Contents

The executable code that was used in compiling the assignment is available on GitHub at https://github.com/maxmheinze/spatial.

Task A

Preliminaries

First, we load the `MASS` package and check what variables there are in the `Boston` dataset.

```
# Header ----------------------------------------------------------------

rm(list = ls())
gc()

pacman::p_load(MASS)

# Check Column Names ----------------------------------------------------

colnames(Boston)
```

Creating the Function

Next, we create the desired function.

```
# Create the function ---------------------------------------------------

boston_quick_ols <- function(dependent, ...) {
    require(MASS)

    # Create a formula string from the inputs
    independents <- paste(c(...), collapse = " + ")
    formula_string <- paste(dependent, "~", independents)

    # Fit the model
    fitted_model <- lm(as.formula(formula_string), data = Boston)

    # Get the summary
    fitted_model_summary <- summary(fitted_model)

    # Get point estimates and confidence intervals
    list_coef <- fitted_model_summary$coefficients
    list_conf <- confint(fitted_model, level = 0.95)
    list_ervr <- fitted_model_summary$sigma^2

    # Output a list
    return(list(coefficients = list_coef[, 1], error_variance = list_ervr, test_statistic_t
    = list_coef[,
        3], test_statistic_p = list_coef[, 4], confidence_intervals = list_conf))

}
```

A Simple Linear Model

Next, we apply the function, using a collection of four independent variables. We predict `medv`, the median home value variable given in the dataset, based on the average number of rooms `rm`, the proportion of homes built prior to 1940 `age`, the distance mean to five Boston employment centers `dis`, and the concentration of nitrogen oxides `nox`. We get the following results:

```
boston_quick_ols("medv", "rm", "age", "dis", "nox")
```

```
## $coefficients
##  (Intercept)           rm          age          dis          nox
##  -6.61135440   8.00051949  -0.06932587  -1.08526888 -22.10858455
##
```

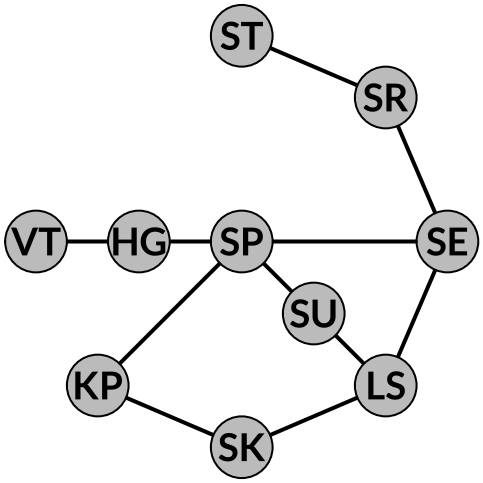
```
## $error_variance
## [1] 37.35166
##
## $test_statistic_t
## (Intercept)          rm         age         dis         nox
##   -1.590287   19.654992   -4.422259   -4.850184   -5.486524
##
## $test_statistic_p
##  (Intercept)           rm          age          dis          nox
## 1.124008e-01 3.520952e-64 1.198751e-05 1.649044e-06 6.516890e-08
##
## $confidence_intervals
##                    2.5 %       97.5 %
## (Intercept) -14.7793094   1.55660058
## rm            7.2007886   8.80025034
## age          -0.1001258  -0.03852595
## dis          -1.5248891  -0.64564866
## nox         -30.0256124 -14.19155674
```

Task B

Creating a Graph and an Adjacency Matrix

We chose the network of all first-district Vienna subway stations. The graph and the adjacency matrix can be found below. Nodes represent individual stations, and edges represent direct subway connections between two stations, without passing another station or changing to another line. We abstract from the existence of different subway lines and from the existence of other stations outside the first district as well as links to these stations. The two-character node labels are to be read as follows: ST is Schottentor, SR is Schottenring, SE is Schwedenplatz, LS is Landstraße, SK is Stadtpark, KP is Karlsplatz, SU is Stubentor, SP is Stephansplatz, HG is Herrengasse, and VT is Volkstheater.

```
# Header ----------------------------------------------------------------
pacman::p_load(igraph, extrafont)


# Create Matrix ---------------------------------------------------------

# Create the adjacency matrix
adj_matrix <- matrix(c(0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
    0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
    1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 0), nrow = 10, byrow = TRUE)

# Define node names
node_names <- c("ST", "SR", "SE", "LS", "SK", "KP", "SU", "SP", "HG", "VT")
dimnames(adj_matrix) <- list(node_names, node_names)

coords_matrix <- matrix(c(0, 0, 3.5, -1.5, 5, -5, 3.5, -8.5, 0, -10, -3.5, -8.5,
    1.75, -6.75, 0, -5, -2.5, -5, -5, -5), ncol = 2, byrow = TRUE)


# Create graph ----------------------------------------------------------

graph_1 <- graph_from_adjacency_matrix(adj_matrix, mode = "undirected")


# Output ----------------------------------------------------------------

plot(graph_1, layout = coords_matrix, vertex.size = 30, vertex.color = "#BBBBBB",
    vertex.label.cex = 1.2, vertex.label.font = 2, vertex.label.family = "Lato",
    vertex.label.color = "black", edge.color = "black", edge.width = 2)
```




```
knitr::kable(adj_matrix)
```

	ST	SR	SE	LS	SK	KP	SU	SP	HG	VT
ST	0	1	0	0	0	0	0	0	0	0
SR	1	0	1	0	0	0	0	0	0	0
SE	0	1	0	1	0	0	0	1	0	0
LS	0	0	1	0	1	0	1	0	0	0
SK	0	0	0	1	0	1	0	0	0	0
KP	0	0	0	0	1	0	0	1	0	0
SU	0	0	0	1	0	0	0	1	0	0
SP	0	0	1	0	0	1	1	0	1	0
HG	0	0	0	0	0	0	0	1	0	1
VT	0	0	0	0	0	0	0	0	1	0

Centrality

Our graph is undirected, and thus the simplest notion of centrality that we can investigate is the nodes' **degree**. We can compute the degree by simply calculating the row sums of the adjacency matrix. The result, given in the table below, is that Stephansplatz is the most central station, having 4 links, and Volkstheater and Schottentor are the least central stations, having one link each.

```
knitr::kable(rowSums(adj_matrix), col.names = c("degree centrality"))
```

	degree centrality
ST	1
SR	2
SE	3
LS	3
SK	2
KP	2
SU	2
SP	4
HG	2
VT	1

Alternatively, we can calculate the nodes' **Eigenvector centrality**. Eigenvector centrality is a centrality measure that takes into account how influential (central) the nodes bordering some node are, where being connected to a more influential node is "rewarded" with a higher centrality measure. To describe the notion mathematically, let N be the set of nodes in the graph from above, and let $H(n)$ be the set of neighbors of some node n (H in this case stands for the *hood*). Let now i and j be two nodes that are in N. We can define a centrality measure c_n such that

$$c_i^{\text{eigenvector}} = \alpha \sum_{j \in H(i)} c_j = \alpha \sum_{j \in N} a_{i,j} c_j,$$

where $a_{i,j}$ is an element of the adjacency matrix and α is some constant. The latter equality follows quite straightforwardly from the fact that $a_{i,j} = 0$ if $j \notin H(i)$ and $a_{i,j} = 1$ if $j \in H(i)$, i.e., the definition of the adjacency matrix. If we now let $\alpha = \frac{1}{\lambda}$, this can be written as

$$\boldsymbol{A}\boldsymbol{c} = \lambda \boldsymbol{c},$$

which means that $\boldsymbol{c}$ is an eigenvector of $\boldsymbol{A}$ corresponding to eigenvalue λ. It follows from the Perron–Frobenius Theorem that there is exactly one (except multiples of itself) eigenvector with all non-negative entries (which is what we desire for the centrality measure) and that it corresponds to the largest of the eigenvalues. Conveniently, the `igraph` package has a function that does the calculations for us. We get the following centrality measures:

```
eigen_centrality(graph_1, directed = FALSE, scale = TRUE, weights = NULL, options =
arpack_defaults()) %>%
    `$`(vector) %>%
    knitr::kable(col.names = "eigenvector centrality")
```

	eigenvector centrality
ST	0.1480101
SR	0.3832787
SE	0.8445071
LS	0.8036100
SK	0.5399797
KP	0.5946913
SU	0.6964970
SP	1.0000000
HG	0.4538490
VT	0.1752621

Again, Stephansplatz is the most central station. Schottentor is now the uniquely least central station.

There exist extensions and variations of eigenvector centrality, such as PageRank centrality, but there are also other approaches. One of these other approaches is the notion of **closeness centrality**, a concept where having shorter average shortest path lengths to all other nodes is rewarded. It is defined as

$$c_i^{\text{closeness}} = \frac{N-1}{\sum_{j \in N, j \neq i} d(i,j)},$$

where $d(\cdot)$ refers to the length of the shortest average path. Again, we are happy to use the implementation the `igraph` package provides to calculate closeness centrality of our stations.

```
knitr::kable(closeness(graph_1), col.names = "closeness centrality")
```

	closeness centrality
ST	0.0333333
SR	0.0454545
SE	0.0625000
LS	0.0526316
SK	0.0454545
KP	0.0500000
SU	0.0500000
SP	0.0666667
HG	0.0476190
VT	0.0344828

Surprise, surprise: Stephansplatz is the most central station and Schottentor is the least central station.

Centrality in a Row-Normalized Network

Row-normalizing means that we divide every element in our adjacency matrix by the corresponding row sum. If we do this, we can see that our adjacency matrix is no longer symmetric:

```
adj_matrix_2 <- adj_matrix/rowSums(adj_matrix)

knitr::kable(round(adj_matrix_2, 2))
```

	ST	SR	SE	LS	SK	KP	SU	SP	HG	VT
ST	0.0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0
SR	0.5	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.0
SE	0.0	0.33	0.00	0.33	0.00	0.00	0.00	0.33	0.00	0.0
LS	0.0	0.00	0.33	0.00	0.33	0.00	0.33	0.00	0.00	0.0
SK	0.0	0.00	0.00	0.50	0.00	0.50	0.00	0.00	0.00	0.0
KP	0.0	0.00	0.00	0.00	0.50	0.00	0.00	0.50	0.00	0.0
SU	0.0	0.00	0.00	0.50	0.00	0.00	0.00	0.50	0.00	0.0
SP	0.0	0.00	0.25	0.00	0.00	0.25	0.25	0.00	0.25	0.0

	ST	SR	SE	LS	SK	KP	SU	SP	HG	VT
HG	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.5
VT	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.0

This means that we can no longer treat our network as an undirected graph, since in-connections and out-connections are differently weighted.

Regarding **degree centrality**, we therefore have to split up our measure into **in-degree** and **out-degree** centrality. Since we normalized row sums, i.e., the sum of outward connections of a node, the out-degree is a somewhat useless measure, since calculating it will always return unity by definition. Calculating the in-degree, however, yields a result:

```
knitr::kable(colSums(adj_matrix_2), col.names = c("in-degree centrality"))
```

	in-degree centrality
ST	0.5000000
SR	1.3333333
SE	1.0833333
LS	1.3333333
SK	0.8333333
KP	0.7500000
SU	0.5833333
SP	1.8333333
HG	1.2500000
VT	0.5000000

A high measure can be interpreted as a station being relatively important for its neighbor stations. If a station has only one neighbor station, and that neighbor station has another station that it neighbors, the first station will have a measure of 0.5. In our example, Stephansplatz is the most central station and Schottentor and Volkstheater are the least central stations.

So, we get the same most and least central stations as in the original degree centrality case. Will that always be the case that stations that are more central in the one measure are also more central in the other? No. For a smooth disproof by counterexample, look at nodes SR and SE.

Our measure of **eigenvector centrality** is affected in so far as that in the original formula,

$$c_i^{\text{eigenvector}} = \alpha \sum_{j \in N} a_{i,j} c_j,$$

$a_{i,j}$ can now assume values between 0 and 1 instead of *just* 0 and 1. Every value that was 0 before is still 0, but all connections that do exist are now weighted by how many other outgoing connections from a station there are. This makes for a slightly different eigenvector:

```
adj_matrix_2 <- adj_matrix/rowSums(adj_matrix)

graph_2 <- graph_from_adjacency_matrix(adj_matrix_2, mode = "directed", weighted = TRUE)



eigen_centrality(graph_2, directed = FALSE, scale = TRUE, weights = NULL, options =
arpack_defaults()) %>%
    `$`(vector) %>%
    knitr::kable(col.names = "eigenvector centrality")
```

	eigenvector centrality
ST	0.5030657
SR	0.6942537
SE	0.8190652

	eigenvector centrality
LS	0.8004645
SK	0.6486233
KP	0.6756410
SU	0.6845443
SP	1.0000000
HG	0.7628555
VT	0.5527755

For our measure of **closeness centrality**, we need a definition of "distance" in a weighted graph. Conventionally, weights are in this case interpreted as the "length" of a node, meaning that a low-weighted connection is related to the notion of two nodes being "closer," and that the distance equals the sum of weights along a path. Since weights *from* a station that has many outgoing connections will be lower (even if incoming connections' weights need not be), having many outgoing connections is rewarded using this measure of centrality with a row-normalized adjacency matrix. We can also see that Stephansplatz gains relatively more compared to the original closeness measure:

```
knitr::kable(closeness(graph_2), col.names = "closeness centrality")
```

	closeness centrality
ST	0.0582524
SR	0.1153846
SE	0.1875000
LS	0.1500000
SK	0.1052632
KP	0.1224490
SU	0.1250000
SP	0.2105263
HG	0.1212121
VT	0.0597015

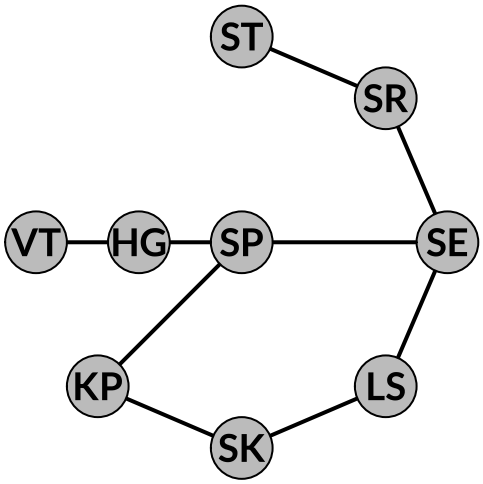
Removing a node

We remove Stubentor because it might be interesting to see what happens if we remove one of Stephansplatz's connections. Let's see:

```
# Header ----------------------------------------------------------------------
pacman::p_load(igraph, extrafont)


# Create Matrix ---------------------------------------------------------------

# Create the adjacency matrix
adj_matrix_3 <- matrix(c(0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
    1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 1, 0), nrow = 9, byrow = TRUE)

# Define node names
node_names_3 <- c("ST", "SR", "SE", "LS", "SK", "KP", "SP", "HG", "VT")
dimnames(adj_matrix_3) <- list(node_names_3, node_names_3)

coords_matrix_3 <- matrix(c(0, 0, 3.5, -1.5, 5, -5, 3.5, -8.5, 0, -10, -3.5, -8.5,
    0, -5, -2.5, -5, -5, -5), ncol = 2, byrow = TRUE)


# Create graph ----------------------------------------------------------------

graph_3 <- graph_from_adjacency_matrix(adj_matrix_3, mode = "undirected")
```

```
# Output ------------------------------------------------------------------
```

```
plot(graph_3, layout = coords_matrix_3, vertex.size = 30, vertex.color = "#BBBBBB",
    vertex.label.cex = 1.2, vertex.label.font = 2, vertex.label.family = "Lato",
    vertex.label.color = "black", edge.color = "black", edge.width = 2)
```




```
knitr::kable(adj_matrix_3)
```

	ST	SR	SE	LS	SK	KP	SP	HG	VT
ST	0	1	0	0	0	0	0	0	0
SR	1	0	1	0	0	0	0	0	0
SE	0	1	0	1	0	0	1	0	0
LS	0	0	1	0	1	0	0	0	0
SK	0	0	0	1	0	1	0	0	0
KP	0	0	0	0	1	0	1	0	0
SP	0	0	1	0	0	1	0	1	0
HG	0	0	0	0	0	0	1	0	1
VT	0	0	0	0	0	0	0	1	0

```
knitr::kable(rowSums(adj_matrix_3), col.names = c("degree centrality"))
```

	degree centrality
ST	1
SR	2
SE	3
LS	2
SK	2
KP	2
SP	3
HG	2
VT	1

```
eigen_centrality(graph_3, directed = FALSE, scale = TRUE, weights = NULL, options =
arpack_defaults()) %>%
    `$`(vector) %>%
    knitr::kable(col.names = "eigenvector centrality")
```

	eigenvector centrality
ST	0.2412297
SR	0.5471942
SE	1.0000000
LS	0.7211592
SK	0.6358438
KP	0.7211592
SP	1.0000000
HG	0.5471942
VT	0.2412297

```
knitr::kable(closeness(graph_3), col.names = "closeness centrality")
```

	closeness centrality
ST	0.0384615
SR	0.0526316
SE	0.0714286
LS	0.0555556
SK	0.0500000
KP	0.0555556
SP	0.0714286
HG	0.0526316
VT	0.0384615

We can see that Stephansplatz suffers and is now exactly as central as Schwedenplatz. Also, there is no longer a difference between Schottentor and Volkstheater as least central stations using any measure of centrality. This makes intuitive sense since the new graph is now symmetrically consisting of a central "circle" and two "appendices" of length two.

Simulation

We say that the coolness factor $\boldsymbol{x}$ is a characteristic of each station. Some stations are cooler and some are less cool. We also assume that coolness $\boldsymbol{x}$ affects the crime rate $\boldsymbol{y}$ at a station. However, crime rates at stations are also influenced by neighboring stations' crime rates and coolness factors, in the style of a linear-in-means model:

$$\boldsymbol{y} = \boldsymbol{x}\beta + \boldsymbol{W}\boldsymbol{x}\gamma + \lambda\boldsymbol{W}\boldsymbol{y} + \varepsilon,$$

where $\varepsilon \sim \mathbf{N}(\mathbf{0}, \boldsymbol{I}\sigma^2)$. Letting $\boldsymbol{S} = (\boldsymbol{I} - \lambda\boldsymbol{W})$, this becomes

$$\boldsymbol{S}\boldsymbol{y} = \boldsymbol{x}\beta + \boldsymbol{W}\boldsymbol{x}\gamma + \varepsilon,$$

which we can simulate as shown in the following. For the simulation, we let $\beta = -1$ (cooler stations have less crime), $\gamma = 1$, $\lambda = 0.65$, and $\sigma^2 = 1$.

```
set.seed(1234)

# Set parameters
N <- length(node_names)
beta <- -1
gamma <- 1
lambda <- 0.65
sigmasquared <- 1


reps <- 1000
estims <- vector("numeric", reps)
```

```
for (i in 1:reps) {

    # Create the coolness vector
    x <- rnorm(length(node_names))
    names(x) <- node_names

    # Rename the adj. matrix W
    W <- adj_matrix_2

    errs <- rnorm(N, 0, sigmasquared)

    Wx <- W %*% x

    # Calculate S = (I - \lambda W)
    S = diag(N) - lambda * W

    # Solve for y (the crime variable)
    y = solve(S, Wx * gamma + x * beta + errs)

    # Fit a linear model
    model_1 <- lm(y ~ x)

    # Store fitted estimates
    estims[i] <- coef(model_1)["x"]
}
```

We then fit the following linear model

$$\boldsymbol{y} = \boldsymbol{x}\beta + \boldsymbol{\varepsilon},$$

and store the coefficients for β.

```
avg_estimate <- mean(estims)
print(avg_estimate)
```

```
## [1] -0.9238921
```

The average β coefficient of 1,000 simulations was -0.9238921, and we conclude the estimate is downward biased in magnitude, since the true value of β was -1.

Task C

Preliminaries

We load the relevant packages for manipulating spatial data.

```
pacman::p_load(dplyr, ggplot2, sf, data.table, readr, eurostat, RColorBrewer, viridis)
```

We download a shapefile for NUTS2 region and a dataset that includes fertility rates at the same regional level.

```
# uncomment in order to download the files anew

# temp <- tempfile(fileext = '.zip')

#
download.file('http://ec.europa.eu/eurostat/cache/GISCO/distribution/v2/nuts/download/ref-nuts-2021-03m
# temp)

# outDir <- './assignment1/data' unzip(temp, exdir=outDir)
# unzip('./data/NUTS_RG_03M_2021_4326_LEVL_2.shp.zip', exdir=outDir)

# Data of interest at NUTS 2 level (Regional fertility rate)
EurostatTOC <- get_eurostat_toc()
df1 <- get_eurostat("tgs00100", time_format = "raw")
```

```
## indexed 0B in  0s, 0B/sindexed 1.00TB in  0s, 1.82PB/s
```

Shapefile CRS

Now we read the shapefile and find out which Coordinate Reference System (CRS) it uses.

```
shp1 <- st_read(dsn = "./assignment1/data", layer = "NUTS_RG_03M_2021_4326_LEVL_2")
```

```
## Reading layer `NUTS_RG_03M_2021_4326_LEVL_2' from data source
##   `/Users/heinzemax/Documents/GitHub/spatial/assignment1/data'
##   using driver `ESRI Shapefile'
## Simple feature collection with 334 features and 9 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -63.15176 ymin: -21.38696 xmax: 55.83509 ymax: 80.83426
## Geodetic CRS:  WGS 84
```

```
# Find out CRS of the shapefile
CRS <- st_crs(shp1)
print(CRS)
```

```
## Coordinate Reference System:
##   User input: WGS 84
##   wkt:
## GEOGCRS["WGS 84",
##     DATUM["World Geodetic System 1984",
##         ELLIPSOID["WGS 84",6378137,298.257223563,
##             LENGTHUNIT["metre",1]]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433]],
##     CS[ellipsoidal,2],
##         AXIS["latitude",north,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433]],
##         AXIS["longitude",east,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433]],
##     ID["EPSG",4326]]
```

We observe that the shapefile uses a projection WGS 84 (EPSG 4326).

Now we apply a new CRS to the shapefile (EPSG 3752) and remove remote regions so that the projection looks less distorted.

```
# Change CRS to EPSG 3752
shp1 <- st_transform(shp1, st_crs(3752))
# Remove overseas regions
overseas <- c("FRY1", "FRY2", "FRY3", "FRY4", "FRY5", "FRZZ", "PT20", "PT30", "PTZZ",
    "ES70", "ESZZ", "NOOB", "NOZZ")
shp1 <- shp1[!shp1$NUTS_ID %in% overseas, ]
```

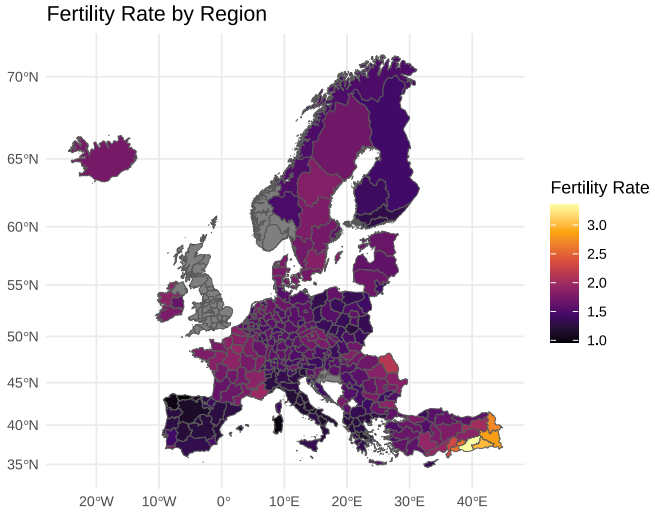
Merging the Shapefile and Creating Two Visualizations

First, we merge the shapefile with the fertility rates dataset. To create the visualizations, we select the fertility rates observations of 2019.

```
df1 <- df1[df1$TIME_PERIOD == 2019, ]

df1 <- df1[!df1$geo %in% overseas, ]

merged_shp1 <- left_join(shp1, df1, by = c(NUTS_ID = "geo"))
```

Now we create the first visualization using a continuous scale.

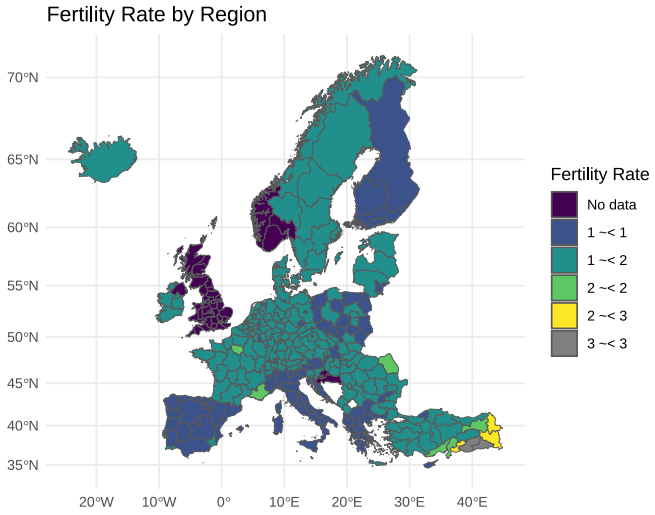
```
# Visualization 1 - Continious scale

ggplot() + geom_sf(data = merged_shp1, aes(fill = values)) + scale_fill_viridis_c(name =
"Fertility Rate",
    option = "inferno") + labs(title = "Fertility Rate by Region") + theme_minimal()
```



For the second visualization, we use a discrete scale with 5 levels. To do that, we cut the values of fertility rate into five segments.

```
# Visualization 2 - Discrete scale

# We cut the values of fertility rate into 5 segments
merged_shp1$value_cat <- cut_to_classes(merged_shp1$values, n = 5)


colors <- viridis(5)
color_na <- ""
ggplot() + geom_sf(data = merged_shp1, aes(fill = ifelse(is.na(value_cat), "No Data",
    as.character(value_cat)))) + scale_fill_manual(name = "Fertility Rate", values =
    c(color_na,
    colors), breaks = c("No Data", levels(factor(merged_shp1$value_cat)))) + labs(title =
"Fertility Rate by Region") +
```

```
    theme_minimal()
```



Different Ways to Store Visualizations

The two main ways to store visualizations are raster graphics and vector graphics. Raster graphics store images as a grid of pixels, where each pixel contains color information. This format is more appropriate for images that contain a high level of detail, with a wide variety of colors and precise shading. On the other hand, vector graphics store images using mathematical equations to define shapes and lines. This storage method is more appropriate for images with less variety of colors and less detail in shading. Since vector files do not use pixels, the images can be scaled without losing sharpness in the details of the shapes.

Accordingly, to store the visualization that uses a continuous scale, it might be more convenient to use raster graphics, since they allow to better display the smooth color gradients. Conversely, to store the discrete-scale visualization, using vector graphics might be a better option since the image contains only solid colors with no smooth gradients. In this way, we could zoom in, rotate the map, or manipulate it in any way without losing any image quality.

We save the first visualization (continuous scale) in a PNG format, which is a raster format. JPEG would be another option. The second visualization (discrete scale) is saved in a PDF format, which uses vector graphics. SVG would be another option here.

```
# ggsave('visualization_1.png', plot = p, device = 'png')
# ggsave('visualization_2.pdf', plot = p, device = 'pdf')
```

Task D

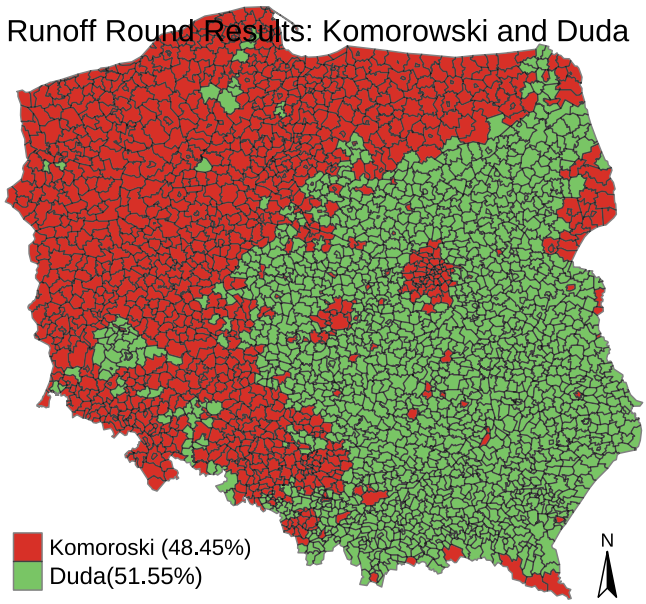
Preliminaries

```
# install required packages install.packages('tmap', repos =
# 'https://r-tmap.github.io/tmap/', type = 'source')
# install.packages('spDataLarge', repos = 'https://nowosad.github.io/drat/',
# type = 'source')


# load the tmap and pDataLarge packages
pacman::p_load("tmap", "spDataLarge", "ggplot2", "spatstat", "spatialreg", "spgwr",
    "spdep", "adehabitatHR", "googleway", "gmapsdistance", "leaflet", "dismo", "raster",
    "sp", "leaflet", "tmap", "foreign", "sf", "terra", "RColorBrewer", "stargazer",
    "rmapshaper", "hrbrthemes")

# load the dataset
data("pol_pres15")
```

Support for Komorowski and Duda

The first choropleth depicts the runoff results between Duda and Komorowski. As observed, Duda is dominant in southeast regions whereas Komorowski is dominant in northwest regions. The total vote share of Duda is 51.55 per cent whereas of that of Komorowski is 48.45 per cent.

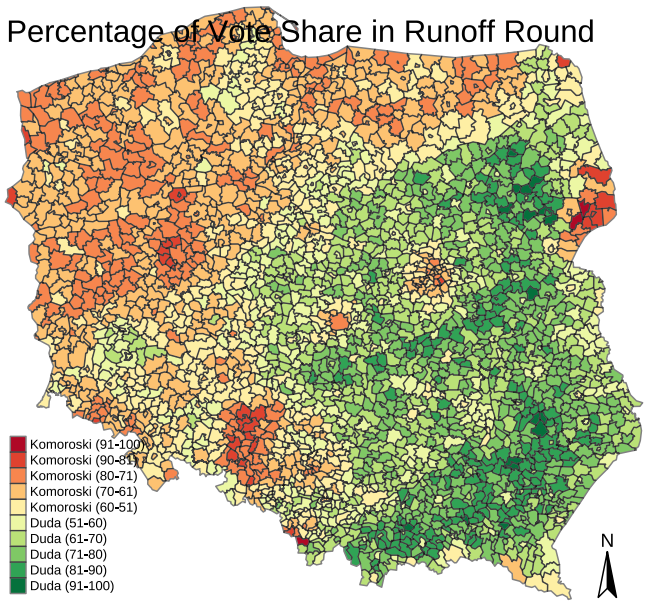
```
# runoff results of komorowski and duda on choropleth
tm_shape(pol_pres15) + tm_fill("II_Duda_share", palette = "RdYlGn", style = "equal",
    n = 2, alpha = 1, breaks = c(0, 0.49, 1), labels = c("Komoroski (48.45%)",
    "Duda(51.55%)")) +
    tm_borders(alpha = 0.5, col = "black") + tm_compass() + tm_layout(title = "Runoff Round
    Results: Komorowski and Duda",
    legend.text.size = 1, legend.title.size = 0.01, legend.position = c("left", "bottom"),
    legend.title.color = "white", frame = FALSE)
```



The second choropleth basically depicts the percentage of vote share in each of the constituences. As the shade of the colour gets lighter for red or green, the vote share for each candidate decreases or conversely for the opposite candidate increases. The colour shades for each candidate is divided into five intervals indicating a consitutency won by a candidate is possible only when it has votes more than 50 per cent or above.

```
# percentage of vote share by each candidate
tm_shape(pol_pres15) + tm_fill("II_Duda_share", palette = "RdYlGn", style = "equal",
```

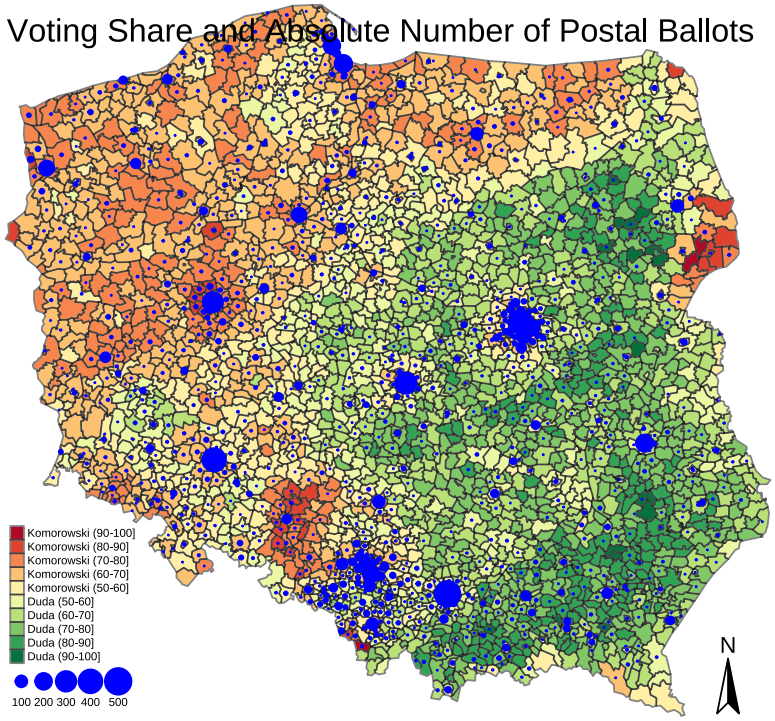
```
    n = 10, breaks = c(0, 0.49, 1), labels = c("Komoroski (91-100)", "Komoroski (90-81)",
        "Komoroski (80-71)", "Komoroski (70-61)", "Komoroski (60-51)", "Duda (51-60)",
        "Duda (61-70)", "Duda (71-80)", "Duda (81-90)", "Duda (91-100)")) + tm_borders(alpha
        = 0.5,
    col = "black") + tm_compass() + tm_layout(title = "Percentage of Vote Share in Runoff
    Round",
    legend.text.size = 0.55, legend.title.size = 0.01, legend.position = c("left",
        "bottom"), legend.title.color = "white", frame = FALSE)
```



Possible Issues With Postal Voting Envelopes

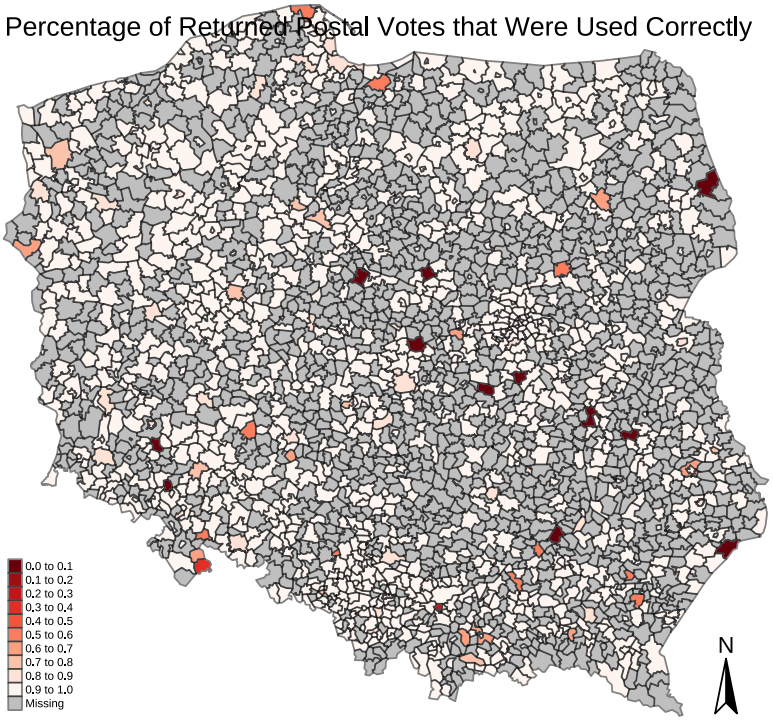
This first graph overlays dots scaled to the number of postal votes over the previous map of relative results. Expectedly, urban areas have the highest absolute amount of postal votes, and as we can see, they also tend to lean Komorowski.

```
tm_shape(pol_pres15) + tm_fill("II_Duda_share", palette = "RdYlGn", style = "equal",
    n = 10, breaks = c(0, 0.49, 1), labels = c("Komorowski (90-100]", "Komorowski (80-90]",
        "Komorowski (70-80]", "Komorowski (60-70]", "Komorowski (50-60]", "Duda (50-60]",
        "Duda (60-70]", "Duda (70-80]", "Duda (80-90]", "Duda (90-100]")) + tm_borders(alpha
        = 0.5,
    col = "black") + tm_dots("II_of_which_voting_papers_taken_from_voting_envelopes",
    col = "blue", shape = 19) + tm_compass() + tm_layout(title = "Voting Share and Absolute
    Number of Postal Ballots",
    legend.text.size = 0.4, legend.title.size = 0.01, legend.position = c("left",
        "bottom"), legend.title.color = "white", frame = FALSE)
```



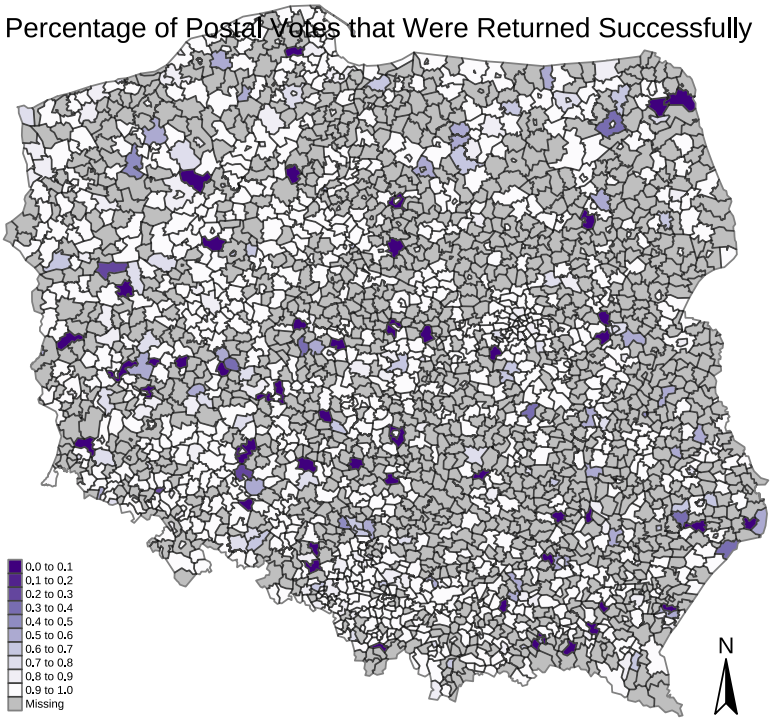
If we look at potential issues regarding postal votes, we can, for example, consider two different "success rates." These are depicted in the following two graphs.

```
pol_pres15_enh <- pol_pres15 %>%
    mutate(II_percentage_postal_voting_successful =
    II_voting_envelopes_placed_in_ballot_box/II_postal_voting_envelopes_received,
        II_percentage_postal_packages_returned =
        II_postal_voting_envelopes_received/II_voters_sent_postal_voting_package)

pol_pres15_enh$II_percentage_postal_voting_successful[is.nan(pol_pres15_enh$II_percentage_postal_voting
<- NA
pol_pres15_enh$II_percentage_postal_packages_returned[is.nan(pol_pres15_enh$II_percentage_postal_packag
<- NA

pol_pres15_enh$II_percentage_postal_voting_successful[is.infinite(pol_pres15_enh$II_percentage_postal_v
<- NA
pol_pres15_enh$II_percentage_postal_packages_returned[is.infinite(pol_pres15_enh$II_percentage_postal_p
<- NA

tm_shape(pol_pres15_enh) + tm_fill("II_percentage_postal_voting_successful", palette =
"-Reds",
    style = "equal", n = 10) + tm_borders(alpha = 0.5, col = "black") + tm_compass() +
    tm_layout(title = "Percentage of Returned Postal Votes that Were Used Correctly",
        legend.text.size = 0.4, legend.title.size = 0.01, legend.position = c("left",
            "bottom"), legend.title.color = "white", frame = FALSE)
```



The first of those two is the percentage of postal votes that were received at the polling stations that were **correctly used and thus could be inserted into the ballot box**. We can see that some municipalities are red, meaning that a high amount of sent back ballot papers could not be counted due to certain irregularities, such as missing signature. We do not observe a clear spatial pattern, except for the fact that "severe" cases seem to predominantly occur in Eastern Poland. Still, we conclude that these issues are on the municipality level and should be investigated by the municipalities marked in red.

```
tm_shape(pol_pres15_enh) + tm_fill("II_percentage_postal_packages_returned", palette =
"-Purples",
    style = "equal", n = 10) + tm_borders(alpha = 0.5, col = "black") + tm_compass() +
    tm_layout(title = "Percentage of Postal Votes that Were Returned Successfully",
        legend.text.size = 0.4, legend.title.size = 0.01, legend.position = c("left",
            "bottom"), legend.title.color = "white", frame = FALSE)
```



The second measure we consider is the percentage of voting packages that were **sent back by the voters**. A low

number means that large percentages of voting packages were sent out, but not received back by the polling station. Again, we observe no spatial pattern, this time, not even an East-West imbalance. Note however that, if those purple outliers are very small municipalities, it could be that low (or even zero) figures could be caused by single individual people requesting and not sending in voting papers.