Assignment 1 - Group 8

Elia Di Gregorio and Robert Auerbach

2024-04-02

Contents

Disclaimer: Available here is the GitHub public repository we used to work on the assignment. Should you have any questions, please do not hesitate to contact: h12039313@s.wu.ac.at and robert.auerbach@s.wu.ac.at.

Exercise A

The dependent variable `medv` shows the median value of owner-occupied homes in $1000s. We chose the following covariates for our linear model:

1) `crim`: per capita crime rate by town;
2) `zn`: proportion of residential land zoned for lots over 25,000 sq.ft;
3) `indus`: proportion of non-retail business acres per town;
4) `chas`: Charles River dummy variable (= 1 if tract bounds river, 0 otherwise);
5) `nox`: nitrogen oxides concentration (parts per 10 million).

Task: Create a function that takes your dependent variable and the covariates as inputs, and return a list with:

– OLS point estimates for the intercept, slope parameters, and the error variance.

– Suitable test statistics with corresponding p-values for the relevant coefficients.

– Intervals of the coefficients for a confidence level of 95%.

```r
# Function to predict property price step-by-step:
pp_pred <- function(dependent_variable, covariates) {

  # Combine the dependent variable and covariates into a data frame
  data <- data.frame(dependent_variable, covariates)

  # Perform OLS regression
  model <- lm(dependent_variable ~ ., data)

  # Extract coefficients
  coefficients <- coef(model)

  # Extract standard errors of coefficients
  se <- summary(model)$coefficients[, "Std. Error"]

  # Calculate t-values
  t_values <- coefficients / se

  # Calculate p-values
  p_values <- 2 * pt(abs(t_values), df = df.residual(model), lower.tail = FALSE)

  # Calculate 95% confidence intervals
  conf_int <- confint.default(model)

  # Store results in a data frame
  results <- data.frame(
    "Coefficients" = coefficients,
    "Standard Errors" = se,
    "t-values" = t_values,
    "p-values" = p_values,
    "Confidence Intervals (95%)" = conf_int
  )
  return(results)
}
```

Call the function and print list:

```r
# Load the Boston dataset
data(Boston)
```

```r
# Selecting a subset of covariates for the model
covariates <- Boston[, 1:5]

# Selecting the dependent variable: property prices
dependent_variable <- Boston$medv

result <- pp_pred(dependent_variable, covariates)
print(result)
```

```
##             Coefficients Standard.Errors  t.values     p.values
## (Intercept)  29.48994059      2.22434765 13.257793 1.365091e-34
## crim         -0.21851904      0.04389177 -4.978588 8.831134e-07
## zn            0.05511047      0.01743801  3.160365 1.671221e-03
## indus        -0.38348055      0.07944258 -4.827141 1.843157e-06
## chas          7.02622266      1.33711744  5.254754 2.198513e-07
## nox          -5.42465902      4.69550757 -1.155287 2.485247e-01
##             Confidence.Intervals..95...2.5.. Confidence.Intervals..95...97.5..
## (Intercept)                      25.13029931                       33.84958187
## crim                             -0.30454533                       -0.13249275
## zn                                0.02093261                        0.08928834
## indus                            -0.53918515                       -0.22777594
## chas                              4.40552064                        9.64692469
## nox                             -14.62768474                        3.77836670
```

Exercise B

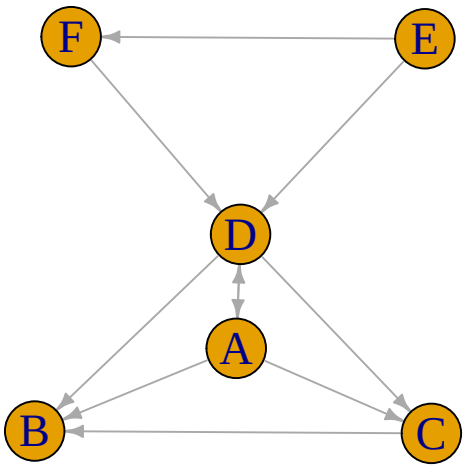
Task: Come up with some network of interest, vaguely related to some real-world example (describe very briefly), with at least six agents and ten edges between them.

Our real world example is one of a supply chain network where there is rarely any reciprocity: construction of airplanes. The main vertices is the firm constructing the airplane (B), where all manufactured parts end up (high in-degree, low out-degree). On the other hand, among manufactures of airplane parts little reciprocity might occur as they need specialized part to finish their own parts (D and A).

The adjacency matrix in R can be drawn as follows:

```
# Define the edges
edges <- c("A", "B", "A", "C", "A", "D", "C", "B", "D", "A",
           "D", "B", "D", "C", "E", "D", "E", "F", "F", "D")

# Create the graph
g <- graph(edges, directed = TRUE)
plot(g, edge.arrow.size = 0.5, vertex.label.cex = 1.5, vertex.size = 30)
```



Printing the adjacency matrix allows us to visualize the graph's structure in a tabular format, making it easier to analyze and understand the connections between vertices.

```
# Get the adjacency matrix
adj_matrix <- as_adjacency_matrix(g)

# Convert the row names and column names to node labels
rownames(adj_matrix) <- V(g)$name
colnames(adj_matrix) <- V(g)$name

# Print the adjacency matrix
print(adj_matrix)
```

```
## 6 x 6 sparse Matrix of class "dgCMatrix"
```

```
##   A B C D E F
## A . 1 1 1 . .
## B . . . . . .
## C . 1 . . . .
## D 1 1 1 . . .
## E . . . 1 . 1
## F . . . 1 . .
```

Who are the most and least central agents in the network? Name, explain, and try to quantify different notions of centrality.

We quantified the in- and out-degree of centrality as well as the eigen-centrality. The in-degree shows how many links are directed towards are node, while the out-degree shows how many links are directed from a node (to another). A node from which a link goes out to another is considered a supplier, a node which is receiving links is considered a buyer.

The degree_table shows how many links go in and out from each node. This way we can quantify the most central buyer and supplier in our network. According to `degree_table` B and D have the most links directed towards them while E has none directed towards it. Hence, B and D are the most central "buyers", E the least. Also, A and D have the most links directed towards others while B has none directed towards others. Hence A and D are the most central "suppliers" and B the least.

```
#Degree Table
id <- cbind(1,3,2,3,0,1)
od <- cbind(3,0,1,3,2,1)

degree_table <- rbind(id, od)

colnames(degree_table) <- c("A","B","C","D","E","F")
rownames(degree_table) <- c("In", "Out")

print(degree_table)
```

```
##     A B C D E F
## In  1 3 2 3 0 1
## Out 3 0 1 3 2 1
```

The measure of eigen-centrality depicts an agent's centrality within a network proportional to the sum of her links to other agents, weighted by their own centrality. D is the most central vertices, while E and F are the least central ones.

```
# Eigen-vector Centrality
ec <- eigen_centrality(g) %>% `$`(vector)
print(round(ec,2))
```

```
##    A    B    C    D    E    F
## 0.88 0.67 0.67 1.00 0.36 0.36
```

How would centrality change if you considered a row-normalized network instead?

We calculate the sums of each row of the adjacency matrix and then divide the adjacency matrix by the row sums. Again, we compute the centrality measures of in- and out-degree. As expected (based on the slides) the out-degree of the agents are equalized to 1. The most central buyers are still B and D.

```
# Compute row sums for normalization
#adj_matrix <- as.data.frame(as.matrix(adj_matrix))
row_sums <- rowSums(adj_matrix, dims = 1)

# Normalize the adjacency matrix
w <- adj_matrix / row_sums

w[is.na(w)] <- 0
```

```r
print(round(w,2))
```

```
## 6 x 6 Matrix of class "dgeMatrix"
##      A    B    C    D E   F
## A 0.00 0.33 0.33 0.33 0 0.0
## B 0.00 0.00 0.00 0.00 0 0.0
## C 0.00 1.00 0.00 0.00 0 0.0
## D 0.33 0.33 0.33 0.00 0 0.0
## E 0.00 0.00 0.00 0.50 0 0.5
## F 0.00 0.00 0.00 1.00 0 0.0
```

```r
# Degree of agents
# Notice that sum_column and sum_row are two functions we self-defined
id <- cbind(sum_column(w, "A"),sum_column(w, "B"),sum_column(w, "C"),
            sum_column(w, "D"),sum_column(w, "E"),sum_column(w, "F"))
od <- cbind(sum_row(w, "A"),sum_row(w, "B"),sum_row(w, "C"),
            sum_row(w, "D"),sum_row(w, "E"),sum_row(w, "F"))

degree_table_norm <- rbind(id, od)

colnames(degree_table_norm) <- c("A","B","C","D","E","F")
rownames(degree_table_norm) <- c("In", "Out")

round(degree_table_norm,2)
```

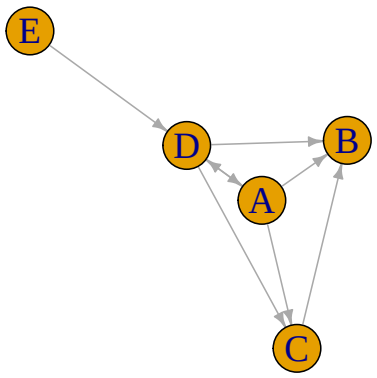
```
##        A    B    C    D E   F
## In  0.33 1.67 0.67 1.83 0 0.5
## Out 1.00 0.00 1.00 1.00 1 1.0
```

How would the network change if you removed or added a specific agent?

We removed agent "F" and repeated the steps from above.

```r
# Re-Define the edges
edges <- c("A", "B", "A", "C", "A", "D", "C", "B",
           "D", "A", "D", "B", "D", "C", "E", "D")

# Re-Create the graph
g1 <- graph(edges, directed = TRUE)
plot(g1, edge.arrow.size = 0.5, vertex.label.cex = 1.5, vertex.size = 30)
```



The most central buyer is now B alone, E still is the least central buyer. The most central supplier is still A and D, the least is still B.

```
# Get the adjacency matrix
adj_matrix1 <- as_adjacency_matrix(g1)

# Convert the row names and column names to node labels
rownames(adj_matrix1) <- V(g1)$name
colnames(adj_matrix1) <- V(g1)$name

# Print the adjacency matrix
print(adj_matrix1)
```

```
## 5 x 5 sparse Matrix of class "dgCMatrix"
##   A B C D E
## A . 1 1 1 .
## B . . . . .
## C . 1 . . .
## D 1 1 1 . .
## E . . . 1 .
```

```
id_g1 <- cbind(1,3,2,2,0)
od_g1 <- cbind(3,0,1,3,1)

degree_table_g1 <- rbind(id_g1, od_g1)

colnames(degree_table_g1) <- c("A","B","C","D","E")
rownames(degree_table_g1) <- c("In", "Out")

print(degree_table_g1)
```

```
##     A B C D E
## In  1 3 2 2 0
## Out 3 0 1 3 1
```

The vertices with the largest eigen-centrality is still D, but now only E has the smallest eigen-centrality.

```
ec_g1 <- eigen_centrality(g1) %>% `$`(vector)
print(round(ec_g1,2))
```

```
##    A    B    C    D    E
## 0.95 0.74 0.74 1.00 0.27
```

For a better comparison of the Reciprocity and Transitivity measures we can observe the following table:

```
rec_g <- reciprocity(g)
trans_g <- transitivity(g)

rec_g1 <- reciprocity(g1)
trans_g1 <- transitivity(g1)

rec <- cbind(rec_g,rec_g1)
trans <- cbind(trans_g, trans_g1)

rectrans_table <- rbind(rec, trans)

colnames(rectrans_table) <- c("Network Complete","Network Removed")
rownames(rectrans_table) <- c("Reciprocity", "Transivity")

round(rectrans_table,2)
```

```
##             Network Complete Network Removed
## Reciprocity             0.20            0.25
## Transivity              0.71            0.80
```

The increase in reciprocity after removing a specific agent suggests that this agent might have had connections that were not reciprocated before its removal. As a result the network has now a higher share of reciprocated links relative to the amount of possible reciprocated links

The increase in clustering coefficient, measured by transitivity, indicates that this agent might have been a bridge between different clusters or groups in the network, but these connections were not necessarily reciprocated. By removing this agent, there is a 10% higher probability that vertices are connected. This is especially true for a network with low reciprocity such as ours.

Task: for our real world example simulating some agent characteristic, we choose level of experience as the agent characteristic and number of defects as the response variable. The coefficient beta we choose to be -0.8 indicating that more experience leads to less defects. We assign some values to the other parameters gamma, lambda and sigma squared as indicated in the code.

```
# Assigning values for simulation:
set.seed(1105) # for replicability

# Parameters
lambda <- 0.4  # Parameter for the network effect
beta <- -1     # Coefficient for the direct effect of X
theta <- 0.5   # Coefficient for the network-lagged effect of X
gamma <- 0.8

reps <- 1000
estimate <- vector("numeric", reps)

for (i in 1:reps) {
  x <- rnorm(6)
  W <- adj_matrix
  e <- rnorm(6, 0, 1)
  Wx <- W %*% x
  S = diag(6) - lambda * W
  y = solve(S, Wx * gamma + x * beta + e)
  x <- as.matrix(x)
  y <- as.matrix(y)
  model_1 <- lm(y ~ x)
```

```
  estimate[i] <- coef(model_1)["x"]
}
estimate_mean <- mean(estimate)

print(round(estimate_mean,2))
```

```
## [1] -1.22
```

```
for (i in 1:reps) {
  x <- rnorm(6)
  W <- adj_matrix
  e <- rnorm(6, 0, 1)
  Wx <- W %*% x %*% theta
  S = diag(6) - lambda * W
  y = solve(S, Wx * gamma + x * beta + e)
  x <- as.matrix(x)
  y <- as.matrix(y)
  model_1 <- lm(y ~ x)
  estimate[i] <- coef(model_1)["x"]
}

estimate_mean_new <- mean(estimate)
print(round(estimate_mean_new,2))
```

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

The linear model estimate depicts a larger negative relationship between level of experience and number of defects than we initially set (-1). Hence, there is a negative bias present. Accounting for θ, the coefficient for the network-lagged effect of X, helps overcoming this bias.

Exercise C

In this exercise, we will work with spatial projections of Europe's NUTS-2 regions. Europe counts 331 subregions, however we excluded overseas territories and focused on the main continental area to improve the final visualization.

Different projections

We accessed its spatial data directly from the GISCO source via `get_eurostat_spatial()` function. By default this function loads the EPSG-4326 projection of the map, corresponding to the World Geodetic System 1984 ensemble (WGS84). It is based on a geocentric datum, meaning it defines the Earth's shape as an ellipsoid (a flattened sphere) rather than a perfect sphere.

```
Europe <- get_eurostat_geospatial(
  resolution = "01",
  nuts_level = 2,
  year = 2021) %>%
  filter(!NUTS_ID %in% c("FRY1", "FRY2", "FRY3", "FRY4", "FRY5", "FRZZ",
                         "PT20", "PT30", "PTZZ", "ES70", "ESZZ", "NOOB", "NOZZ"))

st_crs(Europe)
```

```
## Coordinate Reference System:
##   User input: EPSG:4326
##   wkt:
## GEOGCRS["WGS 84",
##     ENSEMBLE["World Geodetic System 1984 ensemble",
##         MEMBER["World Geodetic System 1984 (Transit)"],
##         MEMBER["World Geodetic System 1984 (G730)"],
##         MEMBER["World Geodetic System 1984 (G873)"],
##         MEMBER["World Geodetic System 1984 (G1150)"],
##         MEMBER["World Geodetic System 1984 (G1674)"],
##         MEMBER["World Geodetic System 1984 (G1762)"],
##         MEMBER["World Geodetic System 1984 (G2139)"],
##         ELLIPSOID["WGS 84",6378137,298.257223563,
##             LENGTHUNIT["metre",1]],
##         ENSEMBLEACCURACY[2.0]],
##     PRIMEM["Greenwich",0,
##         ANGLEUNIT["degree",0.0174532925199433]],
##     CS[ellipsoidal,2],
##         AXIS["geodetic latitude (Lat)",north,
##             ORDER[1],
##             ANGLEUNIT["degree",0.0174532925199433]],
##         AXIS["geodetic longitude (Lon)",east,
##             ORDER[2],
##             ANGLEUNIT["degree",0.0174532925199433]],
##     USAGE[
##         SCOPE["Horizontal component of 3D system."],
##         AREA["World."],
##         BBOX[-90,-180,90,180]],
##     ID["EPSG",4326]]
```

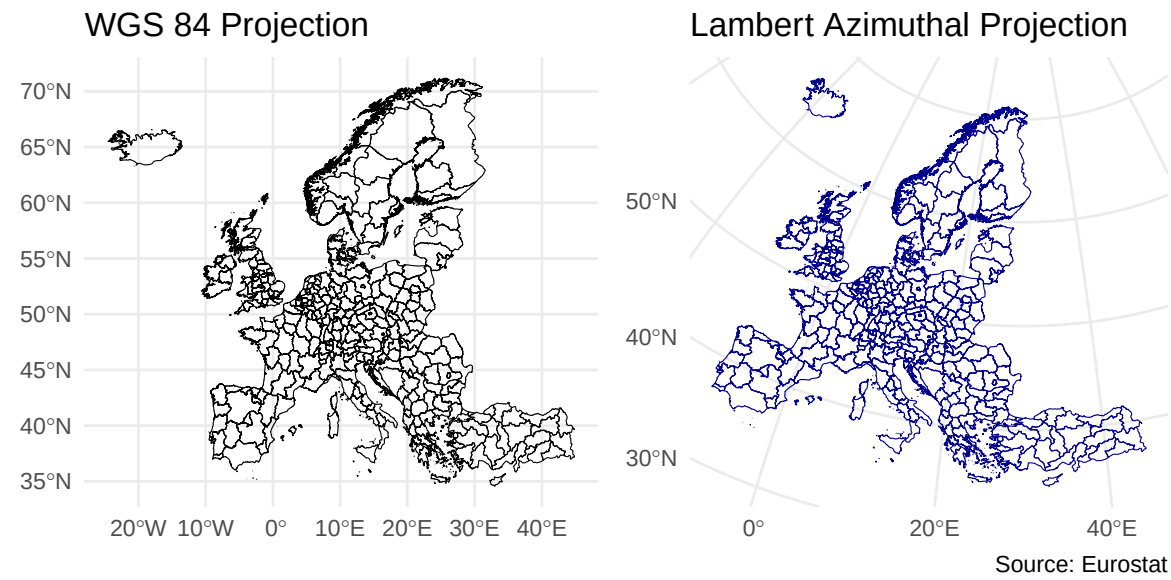
To use another projection we employed the `st_transform()` function and recurred to the Lambert Azimuthal Equal Area Projection. It preserve the relative sizes of areas on the Earth's surface. This means that areas on the map are represented accurately in relation to each other in terms of size, making it suitable for thematic mapping and spatial analysis. However it projects the Earth's surface onto a plane tangent to a specific point (the center of the projection). The difference in projections is highlighted by the following map:

```r
plot_Europe <- ggplot() +
  geom_sf(data = Europe, color = "black", fill = NA) +
  theme_minimal() +
  labs(title = "WGS 84 Projection", size = 12)

Europe_laea <- st_transform(Europe, crs = "+proj=laea +lat_0=45 +lon_0=30")
plot_laea <- ggplot() +
  geom_sf(data = Europe_laea, color = "darkblue", fill = NA) +
  theme_minimal() +
  labs(title = "Lambert Azimuthal Projection", size = 12)

EU_plot_1 <- plot_Europe + plot_laea +
  plot_layout(ncol = 2) +
  labs(caption="Source: Eurostat")

plot(EU_plot_1)
```



Map and Dataset

The dataset used was `tgs00111`: Nights spent at tourist accommodation establishments by NUTS-2 regions (here the Metadata for consultation). The dataset covers internal tourism, in other words tourism flows within the country (domestic tourism) or from abroad to destinations in the country (inbound tourism) for the year 2022. We further enhanced the dataset with two additional columns with the share of domestic and foreign tourist overnights (continuous scale), and one column for a factor variable based on the condition that if the domestic tourism share of overnight stay is greater than 50%, the tourist is labeled as "Domestic Tourist"; otherwise, they are labeled as "Foreign Tourist" (discrete scale).

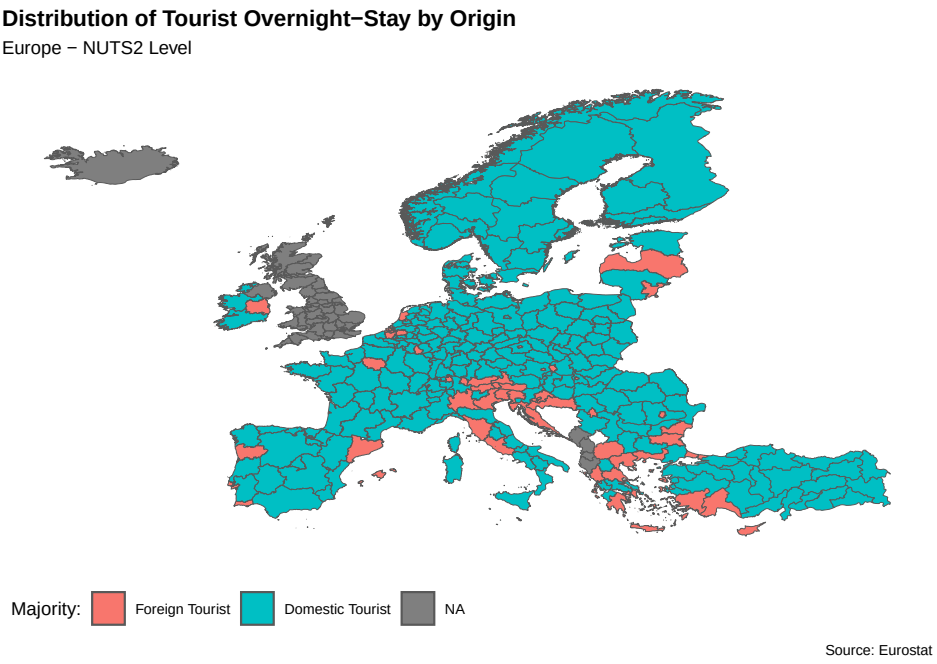
```r
data_nightstay <- get_eurostat("tgs00111",
                               time_format = "raw",
                               filters = list(
                                 TIME_PERIOD = "2022"
```

```
                                )) %>%
  merge(.,Europe, by = "geo", all=TRUE) %>%
  filter(!NUTS_ID %in% c("FRY1", "FRY2", "FRY3", "FRY4", "FRY5", "FRZZ",
                         "PT20", "PT30", "PTZZ", "ES70", "ESZZ", "NO0B", "NOZZ")) %>%
  pivot_wider(., names_from = c_resid, values_from = values) %>%
  mutate(DOM_SHARE = DOM/TOTAL,
         FOR_SHARE = FOR/TOTAL,
         TURIST = factor(ifelse(DOM_SHARE > 0.5, 0, 1), levels = c(1, 0),
                         labels = c("Foreign Tourist", "Domestic Tourist")))
```

By plotting the latter variable, we were able to group together regions based on whether the majority (>50%) of overnight stays in hotels were by domestic tourists or foreign tourists.

```
EU_plot_2 <- ggplot(data_nightstay) +
  geom_sf(aes(fill = TURIST, geometry = geometry)) +
  theme_map() +
  labs(x = NULL, y = NULL,
       title = "Distribution of Tourist Overnight-Stay by Origin",
       subtitle = "Europe - NUTS2 Level",
       caption = "Source: Eurostat")+
  guides(fill = guide_legend(title = "Majority:")) +
  theme(legend.position = "bottom",
        plot.title = element_text(face = "bold"))

plot(EU_plot_2)
```



Not surprisingly, these were the regions with the most famous "Instagrammable" destinations such as:

- Italy: Venice, Milan, Florence, or Rome,
- Spain: Barcelona and the Balearic Islands,
- France: Paris,
- Croatia's coast,
- Austria: ski hubs in the Alpine regions and Vienna,
- the capital region of Brussels, the Netherlands, Portugal, Ireland, and many other Central and Eastern European (CEE) countries,

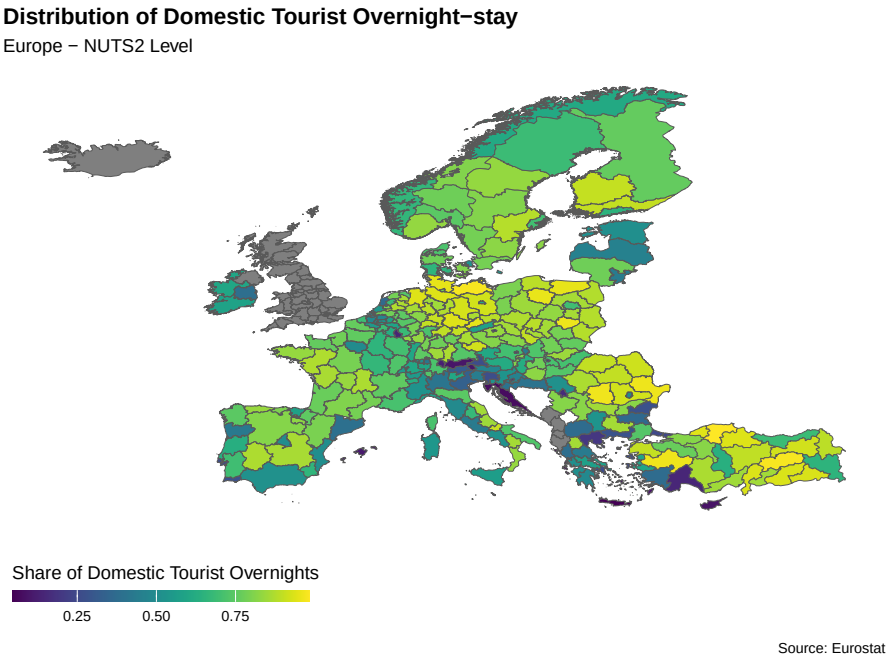
- the coastal regions of Greece and Turkey (Antalya and Bodrum).

Overall, the plot conveys a picture of the main touristic hotspots of Europe where tourists from abroad mostly spend the night, However it might be biased as it does not consider other touristic accommodations (e.g. Airbnb).

The plot for the continuous scale variable focused on the domestic dimension of tourism and represented the share of domestic tourist overnight stays in different European regions.

```
EU_plot_3 <- ggplot(data_nightstay) +
  geom_sf(aes(fill = DOM_SHARE, geometry = geometry)) +
  theme_map() +
  labs(x = NULL, y = NULL,
       title = "Distribution of Domestic Tourist Overnight-stay",
       subtitle = "Europe - NUTS2 Level",
       caption = "Source: Eurostat") +
  theme(legend.position = "bottom",
        plot.title = element_text(face = "bold")) +
  scale_fill_viridis(option = "viridis",
                     direction = 1,
                     name = "Share of Domestic Tourist Overnights",
                     guide = guide_colorbar(direction = "horizontal",
                                            barheight = unit(2, units = "mm"),
                                            barwidth = unit(51, units = "mm"),
                                            draw.ulim = TRUE,
                                            title.position = "top"))

plot(EU_plot_3)
```



Besides stressing the point made by the previous map, this time we get more insights on the distribution of Domestic tourists in Europe. Germany appears to see a higher flow of in-country movement in its Bundesländer. So does some Italian regions of the eastern coast, which are notably more difficult to reach via international connections like Abruzzi and Molise or Calabria. Similarly, we notice a smaller share of foreigner's overnight stay in peripherical CEE countries' regions and Turkey, where the inland regions are mostly hosting domestic tourists.

Storing visualizations

There are two conceptually different ways to store visualizations: raster-based and vector-based formats.

Raster-based formats like PNG and JPEG store images as grids of pixels, making them suitable for complex color gradients and detailed images. However, they are resolution-dependent, which means they can lose quality when scaled up.

On the other hand, vector-based formats such as SVG and PDF store image data using mathematical formulas to define shapes, lines, and colors. This makes them ideal for visualizations with geometric shapes, charts, maps, and illustrations where scalability and high-quality printing are crucial. Unlike raster formats, vector graphics are resolution-independent and can be scaled without loss of quality.

For visualizations created using R and ggplot2, which inherently produce vector graphics, it is recommended to save them in vector-based formats like SVG or PDF. These formats maintain sharpness and clarity when scaled to any size, making them suitable for presentations, printing, and high-resolution displays. SVG is particularly useful for web-based graphics and interactive visualizations, while PDF is excellent for high-quality printing and cross-platform compatibility.

```
# Example on how to save the above-displayed plots:
ggsave("EU_plot_1.svg", plot = EU_plot_1, path = "./plot", device = "svg",
       width = 30, height = 15, units = "cm")
ggsave("EU_plot_2.svg", plot = EU_plot_2, path = "./plot", device = "svg",
       width = 30, height = 15, units = "cm")
ggsave("EU_plot_3.svg", plot = EU_plot_3, path = "./plot", device = "svg",
       width = 30, height = 15, units = "cm")
```

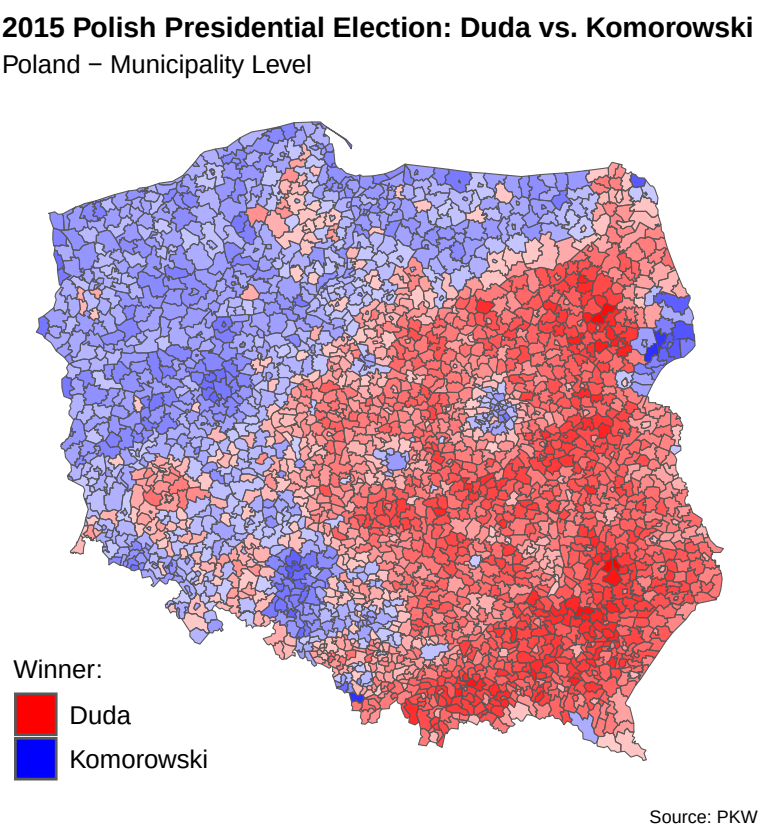
Exercise D

Comparison of support for Komorowski and Duda

```
df <- pol_pres15 %>%
  mutate(Winner = factor(ifelse(II_Duda_share > 0.5, 1, 0), levels = c(1, 0),
                         labels = c("Duda", "Komorowski")),
         Majority_votes_brkdwn =ifelse(II_Duda_share > 0.5, II_Duda_share, II_Komorowski_share))

PL_plot_1 <- ggplot(df) +
  geom_sf(aes(fill = Winner, alpha = Majority_votes_brkdwn, geometry = geometry)) +
  scale_fill_manual(values = c("Duda" = "red", "Komorowski" = "blue")) +
  scale_alpha(range = c(0.2, 1), guide = "none") +
  theme_map() +
  labs(x = NULL, y = NULL,
       title = "2015 Polish Presidential Election: Duda vs. Komorowski",
       subtitle = "Poland - Municipality Level",
       caption = "Source: PKW") +
  guides(fill=guide_legend(title = "Winner:")) +
  theme(
    plot.title = element_text(size = 11, face = "bold"),
    plot.subtitle = element_text(size = 10),
    legend.title = element_text(size = 10),
    legend.text = element_text(size = 10))

plot(PL_plot_1)
```



The provided R code generates a thematic map representing the outcomes of the second round of the 2015 Polish presidential elections at the municipality level. The data is grouped based on whether the winning candidate in each municipality was Duda or Komorowski. The plot uses different shades of red or blue (representing Duda or Komorowski, respectively) to indicate varying levels of majority support for the respective candidate.

The transparency of the colors also varies to reflect the degree of majority/support for the winning candidate in each municipality. Darker shades indicate a higher level of majority/support, while lighter shades represent a lower level of majority/support. This additional dimension helps visually emphasize areas where the winning candidate had a more significant lead in terms of voter share.

Overall, it is possible to see that Duda won by majority in the central and southern-eastern area of the country, whereas Komorowski led the polls in the Western and Northern regions of Poland, besides the main-urban centers.

Postal voting envelopes anomalies

Three types of anomalies are identified:

- `anomaly_invalid`: Checks if there are more invalid voting papers than postal voting envelopes received in either round of voting.
- `anomaly_spelling`: Identifies anomalies related to discrepancies in the number of invalid voting papers and specific errors on postal voting envelopes, such as missing declarations, signatures, voting envelopes, or signs of envelope opening.
- `anomaly_count`: Detects anomalies in the count of voting envelopes placed in the ballot box versus the number of voting papers taken from envelopes in either round of voting.

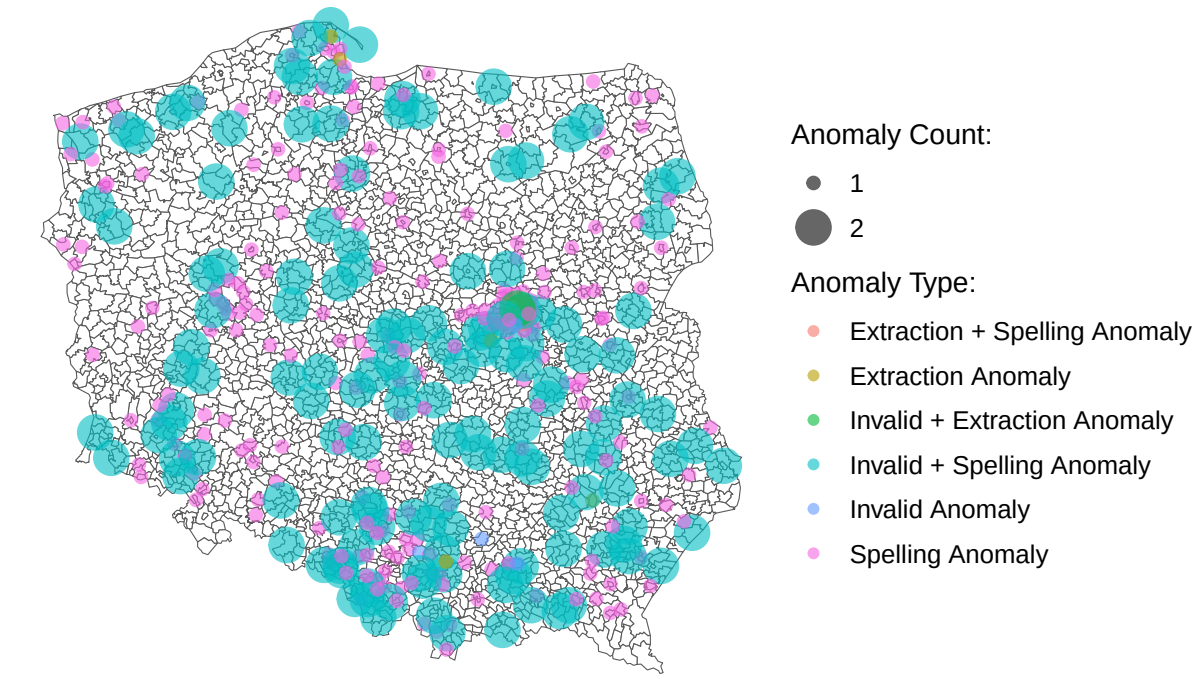
```
data <- pol_pres15 %>%
  mutate(anomaly_invalid = ifelse(I_invalid_voting_papers > I_postal_voting_envelopes_received |
                                  II_invalid_voting_papers > II_postal_voting_envelopes_received, 1,
         anomaly_spelling = ifelse(I_invalid_voting_papers > 0 &
                                   I_PVE_of_which_no_declaration == 0 &
                                   I_PVE_of_which_no_signature == 0 &
                                   I_PVE_of_which_no_voting_envelope == 0 &
                                   I_PVE_of_which_voting_envelope_open == 0 |
                                   II_invalid_voting_papers > 0 &
                                   II_PVE_of_which_no_declaration == 0 &
                                   II_PVE_of_which_no_signature == 0 &
                                   II_PVE_of_which_no_voting_envelope == 0 &
                                   II_PVE_of_which_voting_envelope_open == 0, 1, 0),
         anomaly_count = ifelse(I_voting_envelopes_placed_in_ballot_box != I_of_which_voting_papers_
                                II_voting_envelopes_placed_in_ballot_box != II_of_which_voting_paper
         anomaly = ifelse(anomaly_invalid == 1 | anomaly_spelling == 1 |
                          anomaly_count == 1, anomaly_invalid + anomaly_spelling + anomaly_count, 0)
  mutate(anomaly_type = case_when(
    anomaly_invalid == 0 & anomaly_spelling == 0 & anomaly_count == 0 ~ "None",
    anomaly_invalid == 1 & anomaly_spelling == 0 & anomaly_count == 0 ~ "Invalid Anomaly",
    anomaly_invalid == 0 & anomaly_spelling == 1 & anomaly_count == 0 ~ "Spelling Anomaly",
    anomaly_invalid == 0 & anomaly_spelling == 0 & anomaly_count == 1 ~ "Extraction Anomaly",
    anomaly_invalid == 1 & anomaly_spelling == 1 & anomaly_count == 0 ~ "Invalid + Spelling Anomaly"
    anomaly_invalid == 1 & anomaly_spelling == 0 & anomaly_count == 1 ~ "Invalid + Extraction Anomal
    anomaly_invalid == 0 & anomaly_spelling == 1 & anomaly_count == 1 ~ "Extraction + Spelling Anoma
  ))

# Need x and y coordinates for point, thus convert the geometry column to sf object
data_sf <- st_as_sf(data, wkt = "geometry")
centroid <- st_centroid(data_sf)
data_with_centroid <- cbind(data, st_coordinates(centroid))

PL_plot_2 <- ggplot(data_with_centroid) +
  geom_sf(fill = "white") +
  geom_point(data = subset(data_with_centroid, anomaly > 0), aes(x = X, y = Y, color = as.factor(ano
  theme_void() +
  labs(title = "2015 Polish Presidential Election: anomalies in PVE",
       subtitle = "Poland - Municipality Level",
```

```
       caption = "Data source: PKW",
       color = "Anomaly Type:",
       size = "Anomaly Count:") +
  theme(
    plot.title = element_text(size = 11, face = "bold"),
    legend.text = element_text(size = 10))

plot(PL_plot_2)
```



Anomalies are represented as points `geom_point()` on the map, with each point's color indicating the type of anomaly and size representing the how many anomalies types were overall detected.In this way, one can highlight areas where specific types of anomalies or their combinations were observed more frequently.

By visually exploring anomalies in the handling of voting materials, it appears that this was a widespread problem, involving several municipalities. Hoever, the majority of anomalies seems to have happened in the center-south part of the country which was where Duda won, as highlighted by the previous plot.

Notice that among the three anomalies identified, the one related to spelling relies on a specific assumption. This assumption presumes that all potential reasons for invalidity were documented in the dataset, leading to the anomaly being the higher count of invalid PVEs despite their return with a signed declaration and sealed voting envelope. However, it is crucial to consider that there may be additional, unrecorded issues contributing to this discrepancy, such as misspelling the candidate's name (reason of the variable name). Validating this assumption requires further material, information, or contextual details.

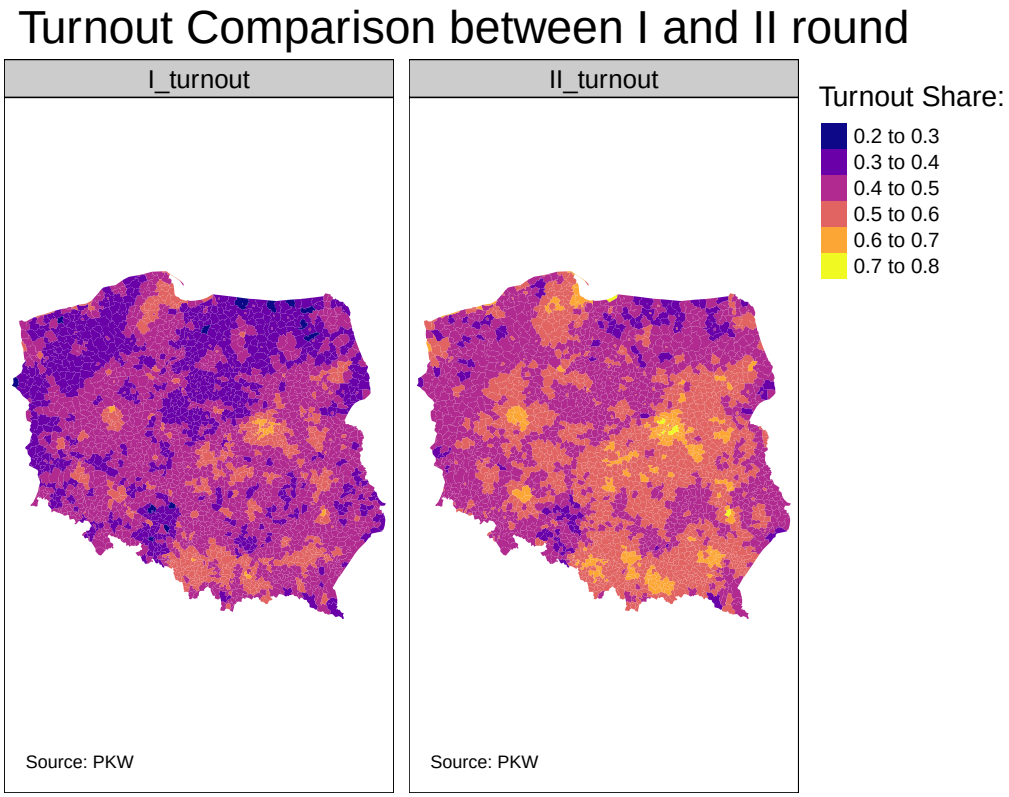
Turnout for each election round

For the final visualization, employing `tm_shape()` function, Turnout share in the first and second round of the presidential election was plotted. We made use of the `tm_facets` specification to combine in a single plot both heatmaps.

```
ds <- pol_pres15 %>%
  pivot_longer(cols = c(I_turnout, II_turnout), names_to = "election", values_to = "turnout")
```

```
PL_plot_3 <- tm_shape(ds) +
  tm_fill(col = "turnout", title = "Turnout Share:", palette = "plasma" ) +
  tm_borders(invisible()) +
  tm_facets(by = "election", free.scales = FALSE) +
  tm_layout(main.title = "Turnout Comparison between I and II round",
            title.position = c("center", "top"),
            frame = TRUE) +
  tm_credits("Source: PKW", position = "left")

PL_plot_3
```



Overall, it appears that in the second round there was a higher public involvement. In rural areas this was less stronger than in the more condensed urban areas, where participation peaked as far as reaching almost 80%.

Additional

```
sessionInfo()
```

```
## R version 4.3.3 (2024-02-29 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22631)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.utf8
## [2] LC_CTYPE=English_United Kingdom.utf8
## [3] LC_MONETARY=English_United Kingdom.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.utf8
##
## time zone: Europe/Rome
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] rsvg_2.6.0        svglite_2.1.3     igraph_2.0.3      glmnet_4.1-8
##  [5] Matrix_1.6-5      MASS_7.3-60.0.1   spDataLarge_2.1.1 ggrepel_0.9.5
##  [9] patchwork_1.2.0   ggthemes_5.1.0    mapproj_1.2.11    maps_3.4.2
## [13] viridis_0.6.5     viridisLite_0.4.2 giscoR_0.4.1      sf_1.0-15
## [17] tmap_3.3-4        eurostat_4.0.0    lubridate_1.9.3   forcats_1.0.0
## [21] stringr_1.5.1     dplyr_1.1.4       purrr_1.0.2       readr_2.1.5
## [25] tidyr_1.3.1       tibble_3.2.1      ggplot2_3.5.0     tidyverse_2.0.0
## [29] pacman_0.5.1
##
## loaded via a namespace (and not attached):
##  [1] DBI_1.2.2          gridExtra_2.3      httr2_1.0.0        tmaptools_3.1-1
##  [5] s2_1.1.6           readxl_1.4.3       rlang_1.1.3        magrittr_2.0.3
##  [9] e1071_1.7-14       compiler_4.3.3     systemfonts_1.0.6  png_0.1-8
## [13] vctrs_0.6.5        wk_0.9.1           shape_1.4.6.1      pkgconfig_2.0.3
## [17] fastmap_1.1.1      backports_1.4.1    labeling_0.4.3     lwgeom_0.2-14
## [21] leafem_0.2.3       ISOweek_0.6-2      geojsonsf_2.0.3    utf8_1.2.4
## [25] rmarkdown_2.26     tzdb_0.4.0         ragg_1.3.0         xfun_0.42
## [29] jsonlite_1.8.8     RefManageR_1.4.0   highr_0.10         terra_1.7-71
## [33] parallel_4.3.3     R6_2.5.1           stringi_1.8.3      RColorBrewer_1.1-3
## [37] cellranger_1.1.0   stars_0.6-4        iterators_1.0.14   Rcpp_1.0.12
## [41] assertthat_0.2.1   knitr_1.45         base64enc_0.1-3    splines_4.3.3
## [45] timechange_0.3.0   tidyselect_1.2.1   rstudioapi_0.16.0  dichromat_2.0-0.1
## [49] abind_1.4-5        yaml_2.3.8         codetools_0.2-19   curl_5.2.1
## [53] lattice_0.22-5     leafsync_0.1.0     plyr_1.8.9         withr_3.0.0
## [57] evaluate_0.23      survival_3.5-8     units_0.8-5        proxy_0.4-27
## [61] xml2_1.3.6         pillar_1.9.0       KernSmooth_2.23-22 foreach_1.5.2
## [65] generics_0.1.3     rprojroot_2.0.4    sp_2.1-3           hms_1.1.3
## [69] munsell_0.5.0      regions_0.1.8      scales_1.3.0       class_7.3-22
## [73] glue_1.7.0         tools_4.3.3        data.table_1.15.2  XML_3.99-0.16.1
## [77] grid_4.3.3         bibtex_0.5.1       crosstalk_1.2.1    colorspace_2.1-0
## [81] raster_3.6-26      cli_3.6.2          rappdirs_0.3.3     textshaping_0.3.7
## [85] fansi_1.0.6        countrycode_1.6.0  gtable_0.3.4       digest_0.6.35
## [89] classInt_0.4-10    farver_2.1.1       htmlwidgets_1.6.4  htmltools_0.5.7
```

```
## [93] lifecycle_1.0.4    leaflet_2.2.2      httr_1.4.7         here_1.0.1
```