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, ...) {

    # 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.

```
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
```

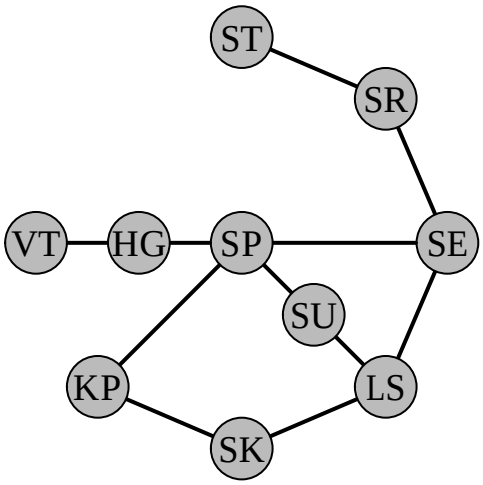
```
## (Intercept)          rm         age         dis         nox
##   -1.590287   19.654992   -4.422259   -4.850184   -5.486524
##
## $test_statistic_p
##  (Intercept)           rm          age          dis          nox
## 1.124008e-01 3.520952e-64 1.198751e-05 1.649044e-06 6.516890e-08
##
## $confidence_intervals
##                    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.

```r
# 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, 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
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
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")
```

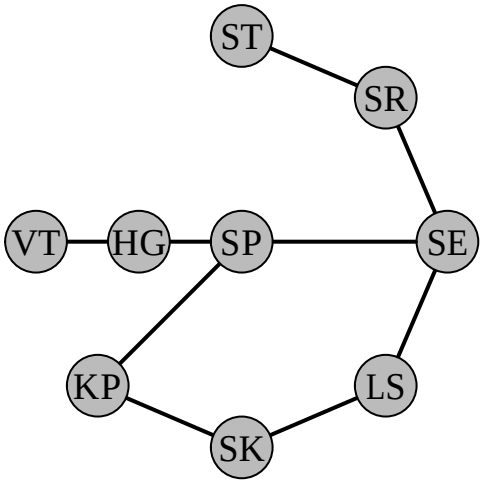
	closeness centrality
ST	0.0582524
SR	0.1153846
SE	0.1875000
LS	0.1500000
SK	0.1052632
KP	0.1224490
SU	0.1250000
SP	0.2105263
HG	0.1212121
VT	0.0597015

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, 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

	degree centrality
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.