The Spatial Equilibrium concept is well known to urban economists. In a nutshell, it states that **in equilibrium there are no rents to be gained by changing locations**.In this post, I want to summarize it briefly following Ed Glaeser presentation and reproduce his 2.1. and 2.2. figures. This is the perfect excuse to play around with the “tidycensus” package.

**Spatial equilibrium in the Alons-Muth-Mills model**

In its simplest form, the model assumes a monocentric closed city. Furthermore, we assume that there are $N$ city dwellers that maximize their utility which depends on consumption and units of land

City dwellers use units of land, consume , and commute to the city center. Consumption is equal to their income, minus transport costs and rental costs per unit of land. City dwellers income is an exogenous wage *w*. Commuting costs are a function of distance and for simplicity we assume it linear . Rental costs per unit of land are , where is the rental cost per unit of land. Thus,

To get to the spatial equilibrium we first rewrite utility function

Then, the first order conditions imply

The equilibrium condition states that rents must decline exactly to offset the increase in transportation cost. The simple assumption that commuting costs are linear implies that the rent gradient is linear:

The key prediction of the model is that land rents will decline with distance to the city center.

**R code**

Let’s take the model to the data and reproduce figures 2.1. of [“Cities, Agglomeration, and Spatial Equilibrium”](https://www.amazon.com/Agglomeration-Spatial-Equilibrium-Lindahl-Lectures/dp/019929044X). The focus are two cities, Chicago and Boston. These cities are chosen because both differ in how easy is to access to their city centers. Chicago is fairly easy, Boston is more complicated. Our model then implies that gradients then should reflect the differential costs to access the city centers.

So let’s begin, the first step is to get some data. To do so I’m are going to use the *“tidycensus”* package. This package will allow me to get data from the census website using their API. We are also going to need the help of three other packages: *“sf”* to handle spatial data, *“dplyr”* my go-to package to wrangle data, and *“ggplot2”* to plot my results.

require("tidycensus", quietly=TRUE)

require("sf", quietly=TRUE)

require("dplyr", quietly=TRUE)

require("ggplot2", quietly=TRUE)

In order to get access to the Census API, I need to supply a key, which can be obtained from <http://api.census.gov/data/key_signup.html>.

census\_api\_key("YOUR CENSUS API HERE")

## To install your API key for use in future sessions, run this function with `install = TRUE`.

Let’s get Median Housing Values from the latest ACS. I’ll use the variable “B25077\_001E” which is the estimated Median Value of Owner-Occupied Housing Units. The package *“tidycensus”* provides the *get\_acs* function that will let me retrieve this variable for the county at the block group level. It is important to note the option *geometry=T*. This option downloads also geometry features of the census block that will allow later on to compute distances.

chicago<-get\_acs(geography = "block group", variables = "B25077\_001E", state = "IL",county = "Cook County",year=2016,geometry = T) #retrieve median housing values for Cook County

boston<-get\_acs(geography = "block group", variables = "B25077\_001E", state = "MA",county = "Suffolk County",year=2016,geometry = T) #retrieve median housing values for Suffolk County

Next, we need the city centers. I create two *“sf”* objects that have the coordinates to the city center.

chicago\_cbd<-st\_as\_sf(x = read.table(text="-87.627800 41.881998"),

coords = c(1,2),

crs = "+proj=longlat +datum=WGS84")

boston\_cbd<-st\_as\_sf(x = read.table(text="-71.057083 42.361145"),

coords = c(1,2),

crs = "+proj=longlat +datum=WGS84")

Now, I need to put everything in the same projection. I re-project my cbds to the same projection as my tibbles.

chicago\_cbd <-chicago\_cbd %>% st\_transform(st\_crs(chicago) )

boston\_cbd <-boston\_cbd %>% st\_transform(st\_crs(boston) )

The next step is to create distances. For that, I use the *st\_distance* function. Given the projection of my tibbles is in meters, I transform all my distances to miles, one meter is 0.000621371 miles.

chicago$dist\_CBD<-st\_distance(chicago,chicago\_cbd) #computes distance to CBD (in meters)

chicago$dist\_CBD<-as.numeric(chicago$dist\_CBD)\*0.000621371 #change units to miles

boston$dist\_CBD<-st\_distance(boston,boston\_cbd) #computes distance to CBD (in meters)

boston$dist\_CBD<-as.numeric(boston$dist\_CBD)\*0.000621371 #change units to miles

The last part is some cleaning up. I add a column with the name of the city, keep block groups in Cook County that are within 10 miles of the city center, and keep the *data.frames* from my tibbles,. Finally, I bind the *data.frames* together.

boston$City<-"Boston"

chicago$City<-"Chicago"

chicago<-chicago %>% filter(dist\_CBD<=10)

chicago<-data.frame(chicago)

boston<-data.frame(boston)

dta<-rbind(chicago,boston)

We are now ready to plot. I use a regular scatter plot with hollow circles and add a linear regression line on a black and white background (*theme\_bw()*)

ggplot(dta, aes(x=dist\_CBD, y=estimate, color=City)) +

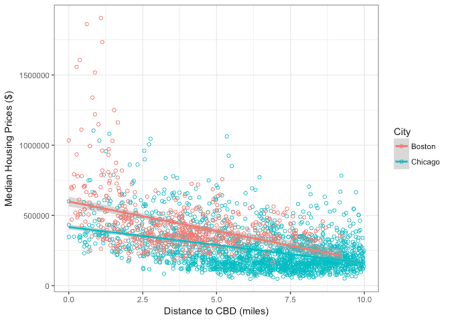
geom\_point(shape=1) + # Use hollow circles

geom\_smooth(method=lm) + # Add linear regression line

xlab("Distance to CBD (miles)") +

ylab("Median Housing Prices ($)") +

theme\_bw()



Conclusion: the data reflect the predictions of the simple Alons-Muth-Mills model, land rents will decline with distance to the city center. The speed at which they decline depends on transportation costs.

**Session Info**

sessionInfo()

R version 3.5.0 (2018-04-23)

Platform: x86\_64-apple-darwin15.6.0 (64-bit)

Running under: macOS High Sierra 10.13.5

Matrix products: default

BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib

LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib

locale:

[1] en\_US.UTF-8/en\_US.UTF-8/en\_US.UTF-8/C/en\_US.UTF-8/en\_US.UTF-8

attached base packages:

[1] stats graphics grDevices utils datasets methods base

other attached packages:

[1] bindrcpp\_0.2.2 ggplot2\_2.2.1 dplyr\_0.7.4 sf\_0.6-3

[5] tidycensus\_0.4.6

loaded via a namespace (and not attached):

[1] Rcpp\_0.12.16 plyr\_1.8.4 pillar\_1.2.2 compiler\_3.5.0

[5] bindr\_0.1.1 class\_7.3-14 tools\_3.5.0 uuid\_0.1-2

[9] digest\_0.6.15 gtable\_0.2.0 jsonlite\_1.5 evaluate\_0.10.1

[13] tibble\_1.4.2 lattice\_0.20-35 pkgconfig\_2.0.1 rlang\_0.2.0

[17] DBI\_0.8 curl\_3.2 rgdal\_1.2-18 yaml\_2.1.18

[21] spData\_0.2.8.3 e1071\_1.6-8 httr\_1.3.1 stringr\_1.3.0

[25] knitr\_1.20 xml2\_1.2.0 hms\_0.4.2 rappdirs\_0.3.1

[29] tigris\_0.7 tidyselect\_0.2.4 classInt\_0.2-3 rprojroot\_1.3-2

[33] grid\_3.5.0 glue\_1.2.0 R6\_2.2.2 foreign\_0.8-70

[37] rmarkdown\_1.9 sp\_1.2-7 readr\_1.1.1 tidyr\_0.8.0

[41] purrr\_0.2.5 udunits2\_0.13 magrittr\_1.5 scales\_0.5.0

[45] backports\_1.1.2 htmltools\_0.3.6 units\_0.5-1 maptools\_0.9-2

[49] assertthat\_0.2.0 rvest\_0.3.2 colorspace\_1.3-2 labeling\_0.3

[53] stringi\_1.1.7 lazyeval\_0.2.1 munsell\_0.4.3 lwgeom\_0.1-4