Modeling Urban Crime Occurrences via Network Regularized Regression

A Simplified Example

Elizabeth Upton and Luis Carvalho

2024-05-08

Introduction and data

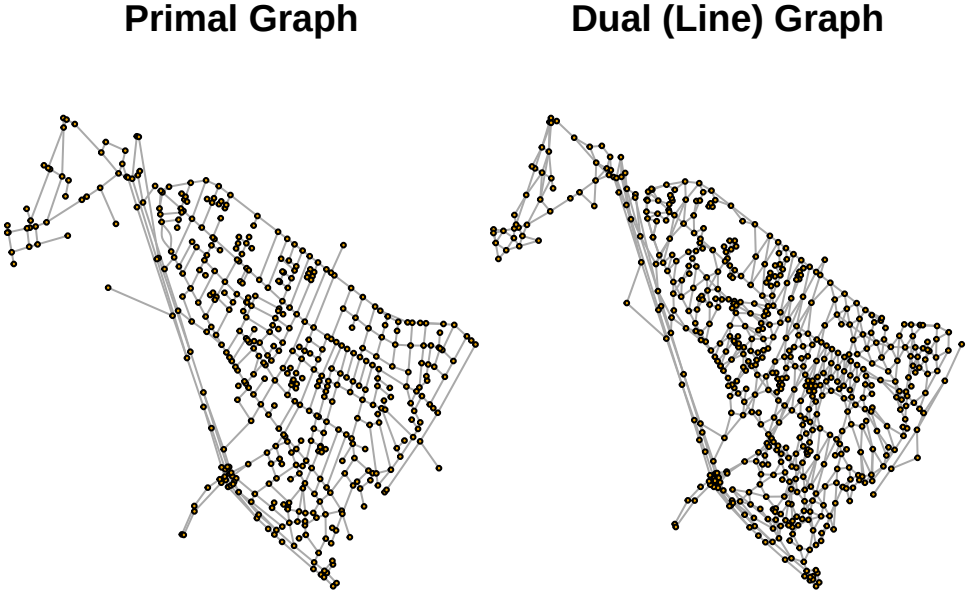
This document illustrates the developed methodologies. The folder `./data` contains the necessary shape files to complete the analysis. For this simplified walkthrough, we focus on one neighborhood in Boston (Charlestown) and one predictor (wealth). We include some code in this pdf output, but refer readers to the .Rmd file for a deeper dive. We developed an R package, `banner`, that includes the functions employed in our analysis. You can find the package files in the folder `./paper_code`. The package will need to be installed locally to execute the code. Throughout this document you will see references to the manuscript where you can find additional details (e.g. paper: sec 2.4).

```
library(tidyverse)
library(sf)
library(tmap)
library(maptools)
library(igraph)
library(banner) #this files for this package exist in ./paper_code

# Load data

epsg_wgs84 <- 4326 # GPS CRS (WGS 84), coordinate system
epsg_ma <- 2249 # NAD83, MA mainland

# [ network ]
boston <- st_read("./data/Zoning_Subdistricts/Zoning_Subdistricts.shp", quiet = TRUE) %>%
  st_transform(epsg_ma) %>%
  # filter to region of interest
  filter(DISTRICT == "Charlestown Neighborhood")

streets <- st_read("./data/Boston_Street_Segments/Boston_Segments.shp") %>%
  st_transform(epsg_ma) %>% filter(lengths(st_intersects(., boston)) > 0) %>%
  # remove bridges, highways, islands, etc:
  filter(ST_TYPE != "BRG" & ST_TYPE != "CSWY" & ST_TYPE != "DM" &
           ST_TYPE != "HWY" & ST_TYPE != "IS")

# [ crime counts ]
crime <- read_csv("./data/CrimeReport.csv") %>%
  filter(OFFENSE_CODE == 520 | OFFENSE_CODE == 521 |
           OFFENSE_CODE == 522) %>% # residential burglaries
```

```
  filter(!is.na(Lat)) %>%
  st_as_sf(coords = c("Long", "Lat")) %>%
  st_set_crs(epsg_wgs84) %>% st_transform(epsg_ma) %>%
  filter(lengths(st_intersects(., boston)) > 0) %>%
  mutate(date = mdy_hm(OCCURRED_ON_DATE)) %>%
  select(-c("OCCURRED_ON_DATE"))

# [ parcels and wealth proxy ]
parcels <- st_read("./data/Parcels2016Data/output.shp") %>% st_transform(epsg_ma) %>%
  filter(lengths(st_intersects(., boston)) > 0) %>% st_buffer(10)

res_parcels <- parcels %>% # residential parcels
  filter(PTYPE < 200 | PTYPE == 903 | PTYPE == 907 | PTYPE == 908) %>% # dorms
  st_buffer(10)
```

Defining the network

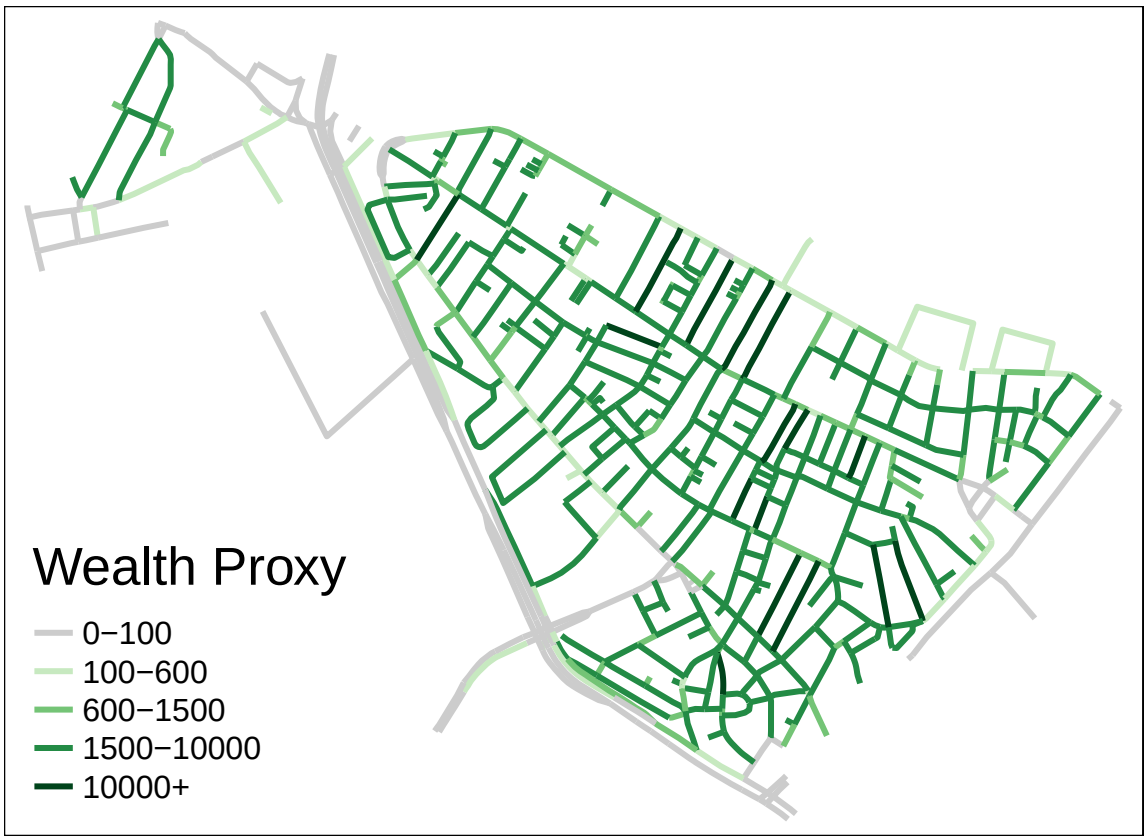
Now we use the information in the shape files to generate the primal and dual graphs.



As a next step, we map the crime occurrences and wealth attributes to their vertices on the dual graph. We include some figures mirroring those in the paper.

```
# [ Crime to Street ]
crime_st_id <- st_nearest_feature(crime, shape)
t <- table(crime_st_id)
Y <- integer(nrow(shape))
Y[as.integer(names(t))] <- t

seg <- data.frame(seg_id, seg_totaldist, Y) #collect all information in seg dataframe

shape <- shape %>%
  mutate(crime_count = Y)

STREET_BUFFER <- 30
```

```r
street_buffer <- shape %>%
  st_buffer(STREET_BUFFER) #buffer around street

street_gross_tax <- st_intersection(res_parcels, street_buffer) %>%
  left_join(as_tibble(res_parcels %>% select(OBJECTID, geometry)), by = "OBJECTID") %>%
  mutate(gross_tax = as.numeric(st_area(geometry.x) / st_area(geometry.y) * AV_TOTAL)) %>%
  select(OBJECTID.1, gross_tax) %>%
  rename(OBJECTID = OBJECTID.1) %>%
  group_by(OBJECTID) %>% summarize(gross_tax = sum(gross_tax))

shape <- shape %>%
  left_join(as_tibble(street_gross_tax %>% select(OBJECTID, gross_tax))) %>%
  select(-geometry.x) %>% # remove polygons
  mutate(gross_tax = ifelse(is.na(gross_tax), 0, gross_tax))

seg$gross_tax <- shape$gross_tax

# [ Residential Buffer: Which streets should we marginalize out of the network? ]
keep <- street_buffer %>%
  filter(lengths(st_intersects(., res_parcels)) > 0)

#Visualizations from the LINE GRAPH perspective
vert <- data.frame(V(gl)$id)
names(vert) <- c("seg_id")
map <- left_join(vert, seg)

V(gl)$crime <- map$Y
V(gl)$gross_tax <- map$gross_tax

# Using map tools (export these to power point to edit)
#wealth
my_pal1 <- c('gray80',"#C7E9C0","#74C476" ,"#238B45","#00441B")
p_w <- tm_shape(shape) +
  tm_lines(col = "gross_tax", palette = my_pal1, size = .025,
           style = "fixed", border.alpha = .5, title.col = "Wealth Proxy",
           breaks = c(0, 100000, 600000, 1500000,  10000000, 500000000),
           labels = c("0-100", "100-600", "600-1500", "1500-10000", "10000+"), lwd= 3) +
  tm_legend( legend.text.size = 1, legend.title.size = 2, legend.width = 20) +
  tm_layout(legend.position = c("left","bottom")); p_w
```




```
#crime
my_pal2 <- c('gray80','yellow','darkorange','red')

p_c <- tm_shape(shape) +
  tm_lines(col = "crime_count", palette = my_pal2,
           breaks = c(0, 1,2,3,5),labels = c("0", "1", "2", "3+"), lwd = 3) +
  tm_legend( legend.text.size = 1, legend.title.size = 2, legend.width = 10) +
  tm_layout(legend.position = c("left","bottom")); p_c
```

We calculate the weights in our matrix based on an inverse of distance between street segment midpoints, marginalize out street segments not in residential areas, pre-process the covariates, and find the needed eigenvalues and eigenvectors of the graph Laplacian.

```
phi <- similarity_phi(dist = E(gl)$dist, median_similarity = 0.8)  #paper: sec. 5.1

sim <- exp(-E(gl)$dist/(max(E(gl)$dist) * phi))

# marginalize out nodes not in residential area:
rem <- dplyr::symdiff(V(gl)$id, keep$OBJECTID)  # These are the ids, not the location
rem2 <- which(V(gl)$id %in% rem == TRUE)

mL <- laplacian_marginal(gl, rem2, weights = sim)
e.mL <- eigen(mL, symmetric = TRUE)

m <- nrow(e.mL$vectors)
e.values <- e.mL$values[m:1]
e.vectors <- e.mL$vectors[, m:1]

# covariates
to_keep <- data.frame(intersect(V(gl)$id, keep$OBJECTID)) %>%
    rename(seg_id = "intersect.V.gl..id..keep.OBJECTID.")

covars <- seg %>%
    right_join(to_keep) %>%
    rename(crime = Y)
```

```r
y <- covars$crime

preds <- data.frame(intercept = 1, log_tax = log(covars$gross_tax +
    1))

preds <- preds %>%
    mutate(log_tax = scale(log_tax))
```

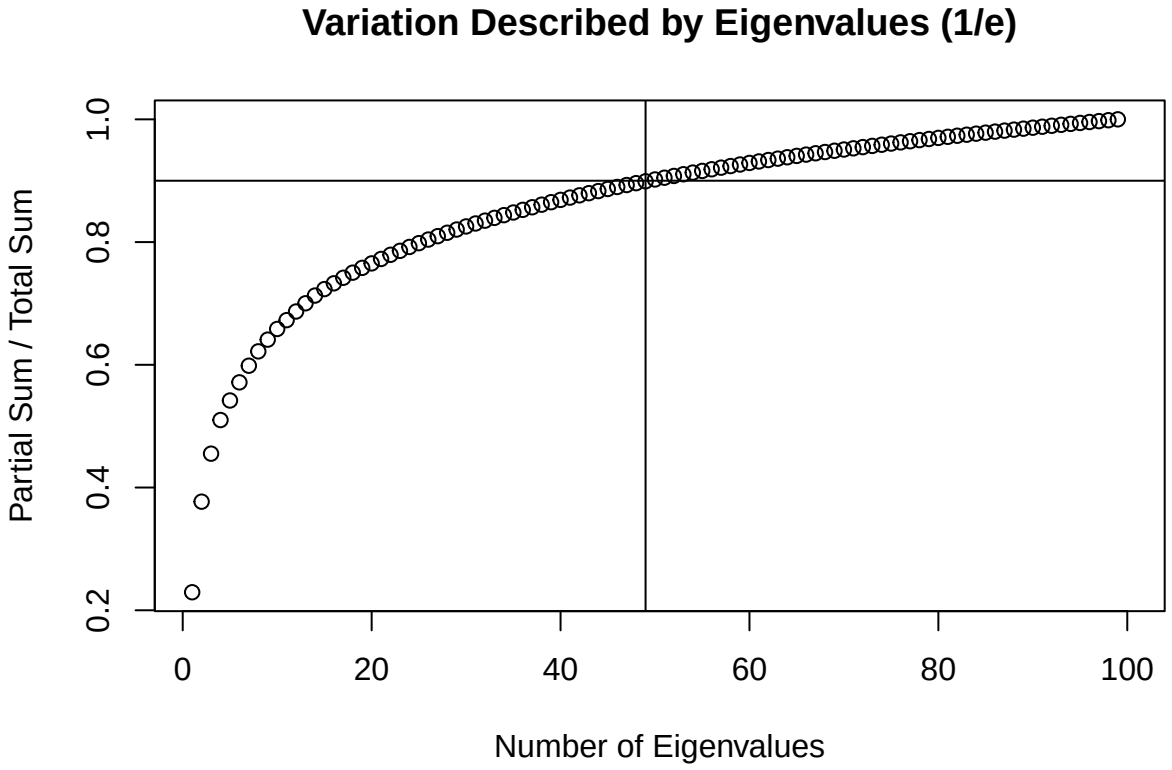
Fitting a model

We first fit a base model with no hot zones (paper: sec 2.2)

```r
# [ create design matrix D_X ]
K <- 100  #max basis expansion, can explore/change
rank <- c(rep(K, ncol(preds)))
X <- expand_design(preds, e.vectors, rank = rank)

# [prior for tau, paper: sec 3.2 and sec 5.2]
perc_var_target <- 0.9
inveigs <- 1/e.values
perc_var <- cumsum(inveigs[2:K])/sum(inveigs[2:K])
(tau_bar <- which.min(abs(perc_var - perc_var_target)))
```

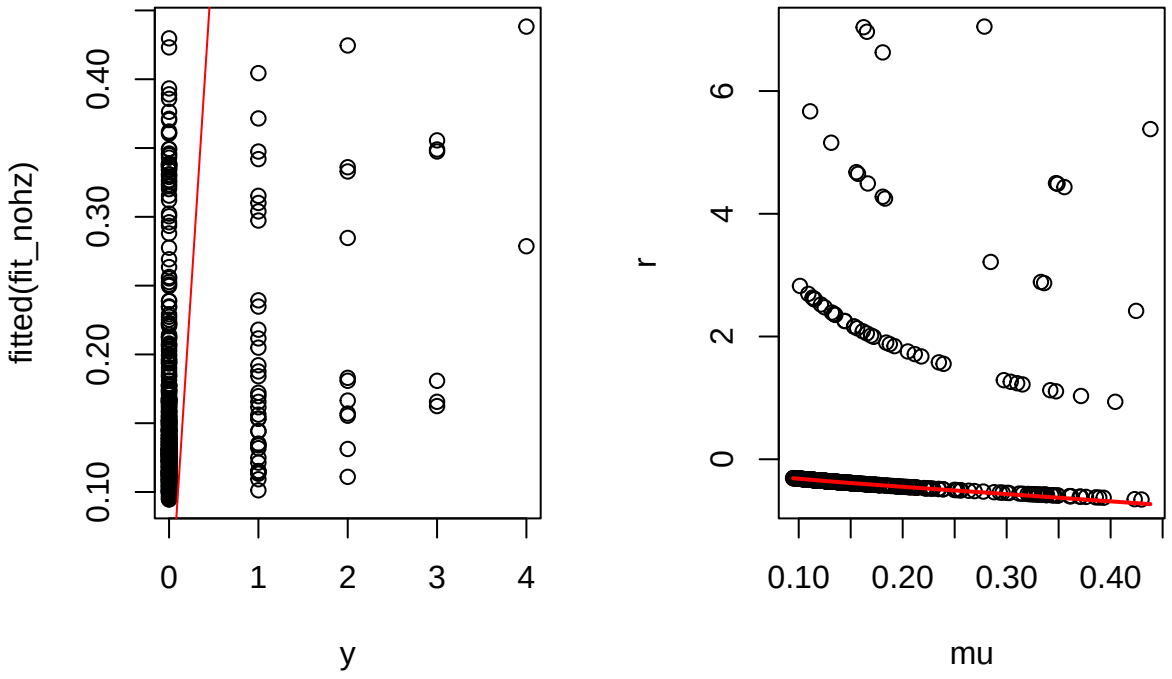
```
## [1] 49
```

```r
plot(perc_var[1:K], xlab = "Number of Eigenvalues", ylab = "Partial Sum / Total Sum",
    main = "Variation Described by Eigenvalues (1/e)")
abline(h = perc_var_target)
abline(v = tau_bar)
```




```r
alpha0 <- 1
alpha1 <- 0.98  #hyper prior params, can explore
log_tau_p <- log(tau_prior(K, tau_bar, alpha0, alpha1))

# [ choose best lambda according to the loop]

# lambda_values <- c(10, 100, 1000) pl <- rep(NA,
# length(lambda_values)); names(pl) <- paste0('l',
# lambda_values) for (j in seq_along(lambda_values)) {
# lambda <- lambda_values[j] res <- em_network_rank(y, X,
# mL, lambda, log_tau_p, control = list(epsilon = 1e-4,
# trace = TRUE)) tau <-
# centroid_network_rank(exp(res$logptau)) X1 <-
# expand_design(preds, e.vectors, rank = tau) fit <-
# try(glm(y ~ X1 - 1, family = res$family, method =
# make_fitter(lambda, X1, mL))) if (class(fit)[1] !=
# 'try-error') pl[j] <- sum(press_loop(fit)) message('[',
# j, '] l = ', lambda, ' | press loop = ', round(pl[j], 2))
# }

lambda_nohz <- 10

# [ECM algo for network basis rank determination, paper:
# sec 3.2 ]
res_nohz <- em_network_rank(y, X, mL, lambda_nohz, log_tau_p,
    control = list(epsilon = 1e-04, trace = TRUE))

# [ sequential centroid estimation for network basis rank ]
tau <- centroid_network_rank(exp(res_nohz$logptau))
```

```
X1 <- expand_design(preds, e.vectors, rank = tau)
fit_nohz <- try(glm(y ~ X1 - 1, family = res_nohz$family, method = make_fitter(lambda_nohz,
    X1, mL), control = list(trace = TRUE)))
op <- par(mfrow = c(1, 2))
plot(y, fitted(fit_nohz))
abline(0, 1, col = "red")
mu <- fitted(fit_nohz)
r <- (y - mu)/sqrt(mu)
plot(mu, r)
lines(lowess(mu, r), col = "red", lwd = 2)
```

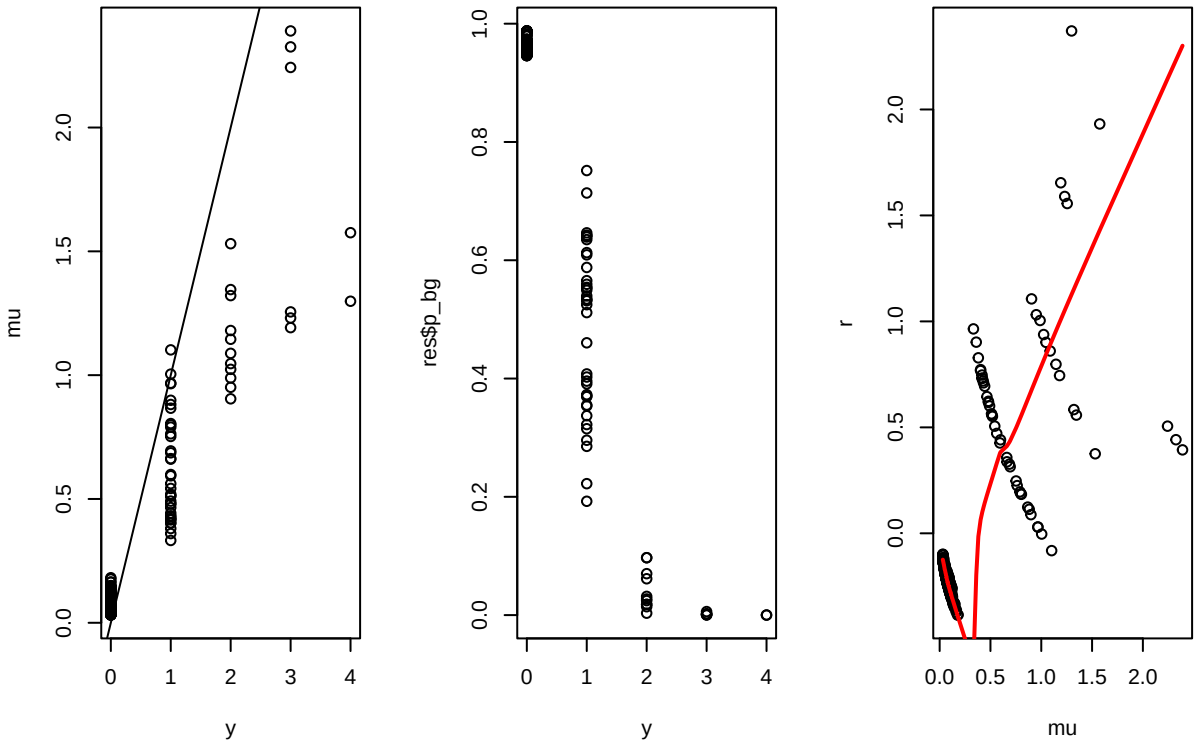



```
par(op)
```

The results don't look good. Let's extend the model and introduce an indicator aimed at capturing hot zones.

```
# [ full model with hot zones, paper: eq 2.4 ]

rank <- c(rep(K, 2))
X <- X_hz <- expand_design(preds, e.vectors, rank = rank)  #D_X, U_X
X_bg <- as.matrix(preds)  # simple, no network expansion, B
p_bg_init <- as.integer(y <= 1)  # simple threshold

# run below code to choose best lambda value according to
# the press loop

# l_values <- c(10, 100, 1000) pl <- matrix(nrow =
# length(l_values), ncol = length(l_values)) for (i in
# seq_along(l_values)) { for (j in seq_along(l_values)) {
# lambda <- l_values[i]; lambda_hz <- l_values[j] res <-
```

```
# em_network_rank_bg(y, X, X_bg, X, mL, lambda, lambda_hz,
# log_tau_p, p_bg_init, control = list(maxit = 20, epsilon
# = 1e-4, trace = TRUE)) tau <-
# centroid_network_rank(exp(res$logptau)) X1 <- X[,
# get_index_rank(attr(X, 'rank'), tau), drop = FALSE]
# tau_hz <- centroid_network_rank(exp(res$logptau_hz))
# X1_hz <- X_hz[, get_index_rank(attr(X_hz, 'rank'),
# tau_hz), drop = FALSE] fit <- em_network_bg(y, X1, X_bg,
# X1_hz, mL, lambda, lambda_hz, p_bg_init, control =
# list(maxit = 50, epsilon = 1e-4, trace = TRUE))
# plot_em_fit(y, X1, X_bg, fit) pl[i, j] <- fit$pl
# message('>>> [', i, ', ', j, '] PL = ', round(pl[i, j],
# 2)) } }


lambda <- 10
lambda_hz <- 10

# [ ECM algo for network basis rank determination and hot
# zones ]
res_em <- em_network_rank_bg(y, X, X_bg, X, mL, lambda, lambda_hz,
    log_tau_p, p_bg_init, control = list(maxit = 50, epsilon = 1e-04,
        trace = TRUE))

tau <- centroid_network_rank(exp(res_em$logptau))
X1 <- X[, get_index_rank(attr(X, "rank"), tau), drop = FALSE]
tau_hz <- centroid_network_rank(exp(res_em$logptau_hz))
X1_hz <- X_hz[, get_index_rank(attr(X_hz, "rank"), tau_hz), drop = FALSE]

fit_em <- em_network_bg(y, X1, X_bg, X1_hz, mL, lambda, lambda_hz,
    p_bg_init, control = list(maxit = 50, epsilon = 1e-04, trace = TRUE))
plot_em_fit(y, X1, X_bg, fit_em)
```




```
message("PL = ", fit_em$pl)
```

Much better! Lastly, we can plot the wealth coefficients (at the count level) across the network to understand how the effects vary smoothly across the neighborhood.

```
# [ removing non residential nodes from shape file ]
shape2 <- shape %>%
    filter(OBJECTID %in% V(gl)$id)
shape2 <- shape2 %>%
    filter(!OBJECTID %in% rem)

# [ calculating beta (at street segment level) ]
shape2$logtax_effect <- drop(e.vectors[, 1:tau[2], drop = FALSE] %*%
    fit_em$coef[tau[1] + 1:tau[2]])

tm_shape(shape2) + tm_lines(col = "logtax_effect", palette = my_pal1,
    size = 0.05, style = "quantile", lwd = 3)
```


logtax_effect
−0.029 to −0.003
−0.003 to −0.000
−0.000 to 0.002
0.002 to 0.005
0.005 to 0.097