R Notebook

```
library(tmap)
library(maptools)
```

```
## Loading required package: sp
```

```
## Checking rgeos availability: FALSE
##      Note: when rgeos is not available, polygon geometry     computations in maptools depend on gpcl
##      which has a restricted licence. It is disabled by default;
##      to enable gpclib, type gpclibPermit()
```

```
library(raster)
library(SpatialRDD)
library(ggplot2)
library(sf)
```

```
## Linking to GEOS 3.7.2, GDAL 2.4.2, PROJ 5.2.0
```

```
library(stars)
```

```
## Loading required package: abind
```

```
library(rgdal)
```

```
## rgdal: version: 1.5-23, (SVN revision 1121)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 2.4.2, released 2019/06/28
## Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/3.6/Resources/library/rgdal/gdal
## GDAL binary built with GEOS: FALSE
## Loaded PROJ runtime: Rel. 5.2.0, September 15th, 2018, [PJ_VERSION: 520]
## Path to PROJ shared files: /Library/Frameworks/R.framework/Versions/3.6/Resources/library/rgdal/proj
## Linking to sp version:1.4-5
## Overwritten PROJ_LIB was /Library/Frameworks/R.framework/Versions/3.6/Resources/library/rgdal/proj
```

```
library(rdrobust)
library(xtable)
```

```
##
## Attaching package: 'xtable'
```

```
## The following object is masked from 'package:maptools':
##
##     label
```

```
library(stargazer)
```

```
##
## Please cite as:
```

```
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.
```

```
##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

```
library(lwgeom)
```

```
## Linking to liblwgeom 3.0.0beta1 r16016, GEOS 3.7.2, PROJ 5.2.0
```

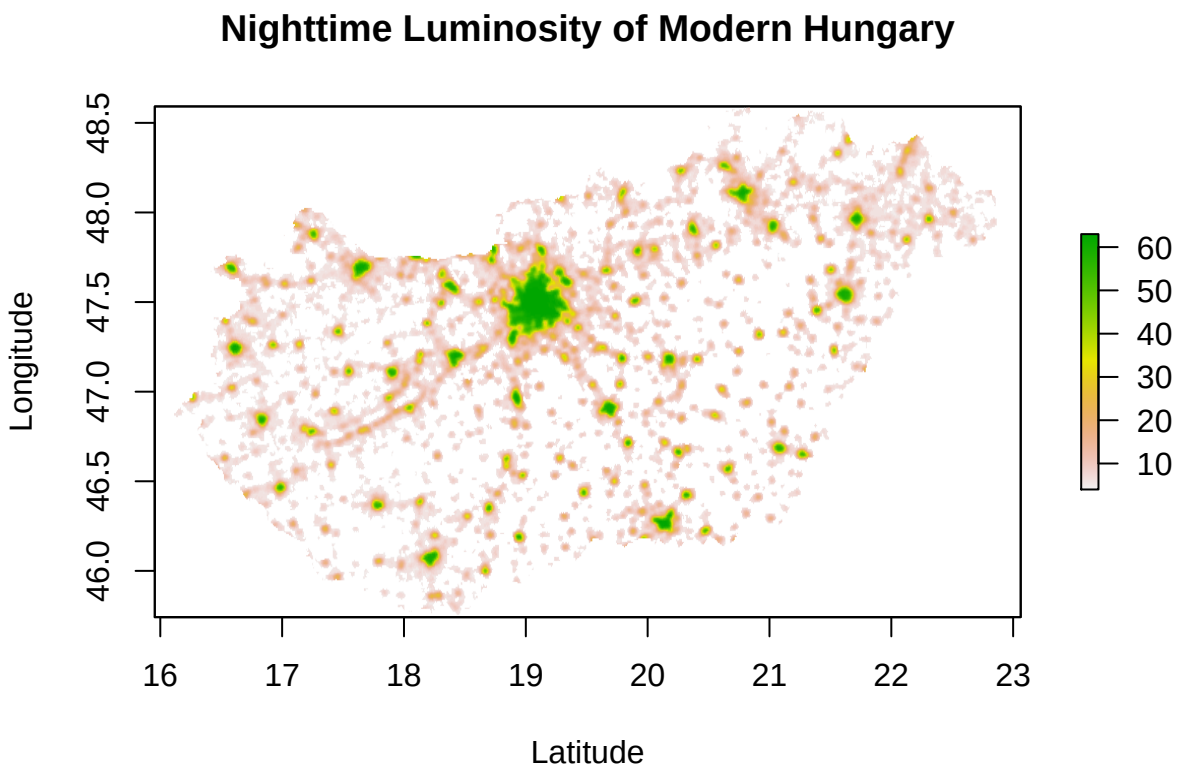
```
library(margins)
```

How long does history leave its mark on economic development? This notebook accompanies a paper exploring that question for the region of transdanubia in modern Hungary using a spatial RDD with NOAA luminosity data. The results suggest a causal link between the area of modern Hungary settled by the Romans and modern economic development in Europe. The concentration of settlement along the right bank, coupled with the continuity between the Roman and modern road systems are highly suggestive of increased economic development on the Roman side, and I hypothesize that continuity between the ancient roads and modern roadways despite the waves of invasions in late antiquity explains the "headstart" given to western Hungary. See the paper for more historical background.

The Roman settlement of Western Hungary is potentially a nice quasi-experiment in the long run impact of infrastructure. I run robust linear regressions, and then the danube boundary is used to run naive RDD with het? eroskedasticity robust errors and data generated bandwidths, and finally fully spatial BDD regressions. Recentered distance to the Danube is used for the running variable in all discontinuity regressions. Since the luminosity data is a skewed distribution of discrete counts, I also run GLM Poisson regressions to predict luminosity levels in the Roman area. The unit of analysis is the 30 arc second pixel. I conduct all preliminary geospatial work in QGIS. The luminosity raster data is clipped to Hungary's modern borders, and a vector shapefile representing the course of the Danube used to cut a line representing the boundary discontinuity, roughly following the center of the river. I create a polygon representing the treatment area of Transdanubia as can be seen in figure 4. I convert each pixel representing luminosity to a point at its centroid. Using the poly? gon representing the treatment area I then generate a binary variable for each luminosity point value. Then I calculate the closest distance between each point and the boundary line, re?centering the distances so that 0 represented the Danube boundary. Figure 6 illustrates each of these steps.

The first thing to do is to load the .tif satellite imagery files. I've already used a GIS program to cut Hungary's modern borders from the global image that the NOAA supplies.

```
hungary <-raster("~/Desktop/DFP/tif files/Hungary_lights_no0.tif")
plot(hungary, main = "Nighttime Luminosity of Modern Hungary", xlab = "Latitude", ylab = "Longitude")
```

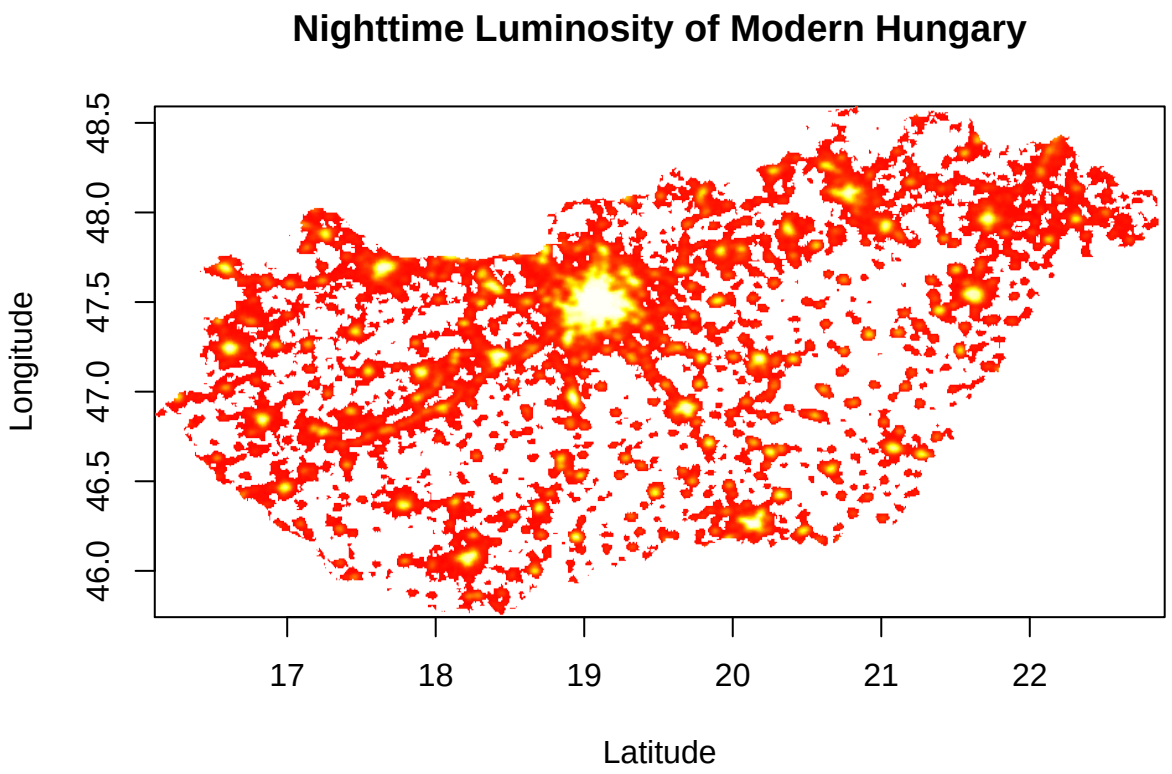


Here's a plot with finer grained binning for the luminosity measure. It'll come in handy later.

```
col = heat.colors(63)
cellStats(hungary, range)
```

```
## [1]  4 63
```

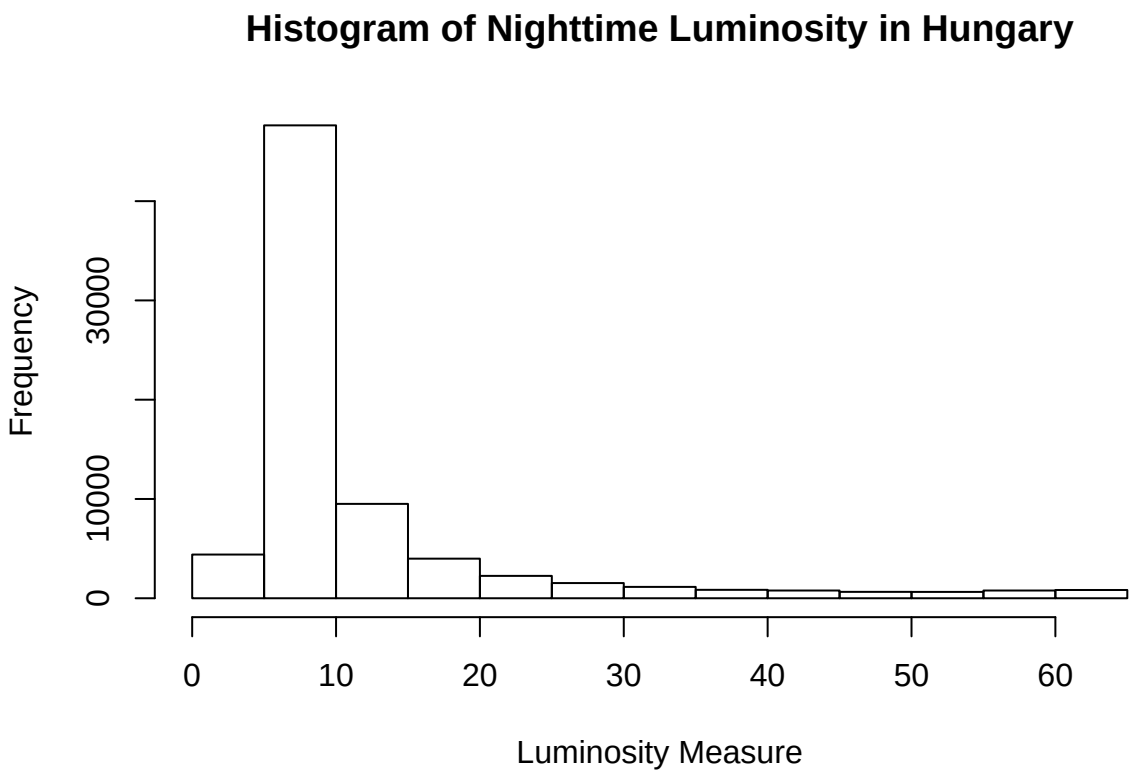
```
image(hungary, col=col, xlab = "Latitude", ylab = "Longitude", main = "Nighttime Luminosity of Modern Hu
```



Looks good. The NOAA measurement is on a 0-100 scale. The main cluster in the center is Budapest, and lake Balaton is visible to the west.

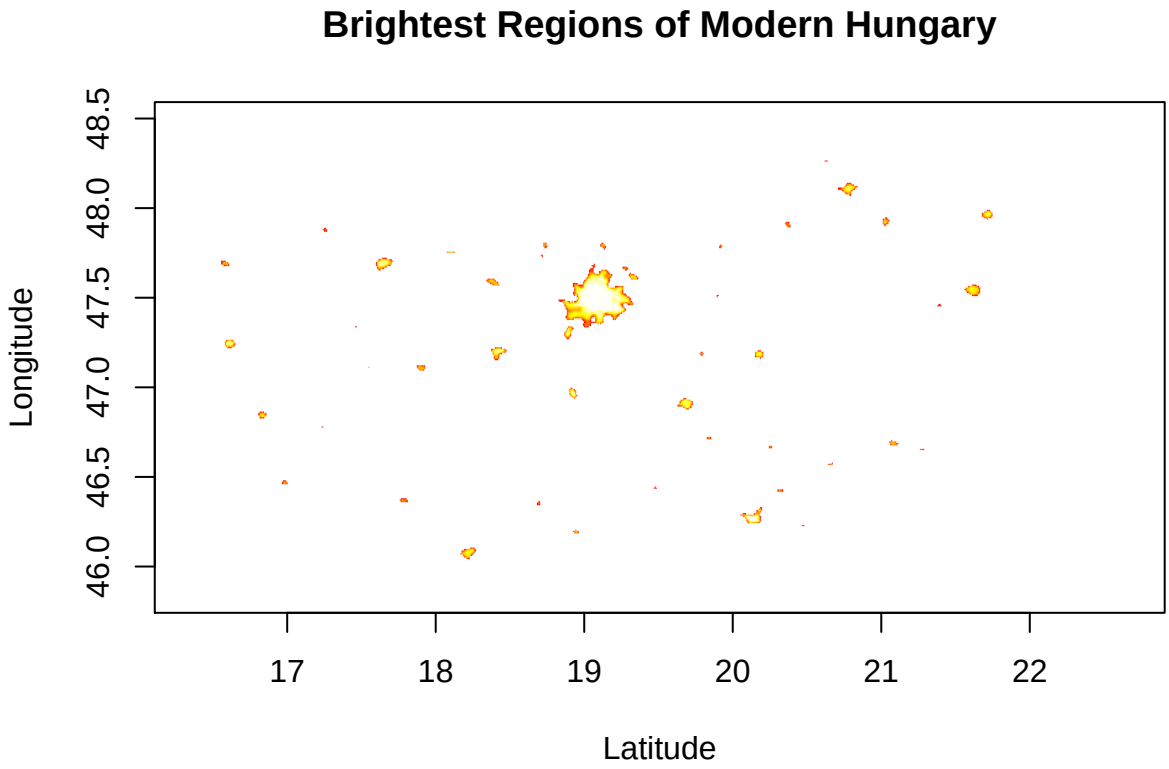
Now for a sense of the distribution of luminosity. You can see below that the vast majority of points have very low light, with a few being much brighter.

```
hungary2 <- as.data.frame(hungary)
hist(hungary2$Hungary_lights_no0, main = "Histogram of Nighttime Luminosity in Hungary", xlab = "Lumino
```



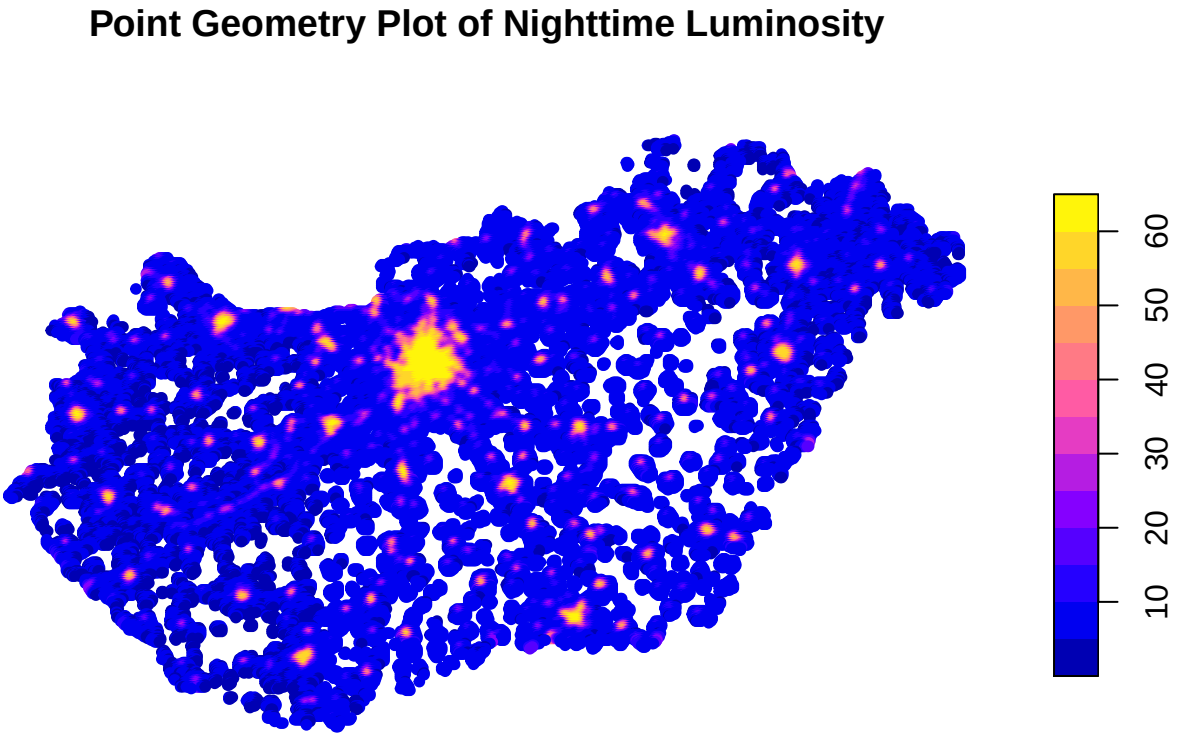
The brightest regions are plotted below.

```
image(hungary, zlim=c(50,63), col=col, main = "Brightest Regions of Modern Hungary", xlab = "Latitude",
```



Now I reload the data as a stars object, to facilitate transformation to an sf object, and converting all the points to point geometries.

```
hungary_stars <- read_stars("~/Desktop/DFP/tif files/Hungary_lights_no0.tif")
hungary.sf <- st_as_sf(hungary_stars, as_points = TRUE, merge = FALSE, long = TRUE, crs = 4326, coords =
plot(hungary.sf, main = "Point Geometry Plot of Nighttime Luminosity", pch = 20)
```



Now for the discontinuity, the Danube river. Remember the whole analysis hinges on using this boundary. I already cut the shape using GIS software.

```
danube <- readOGR(dsn="/Users/andrewfox/Desktop/DFP/danube", layer="discont")
```

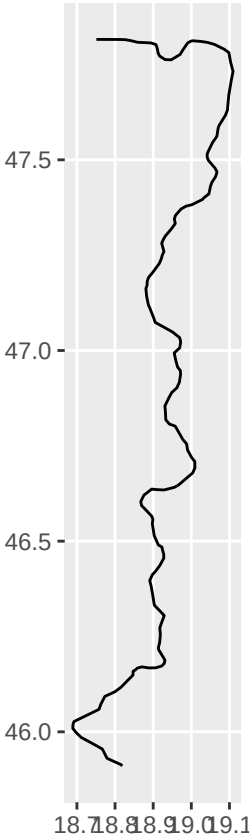
```
## OGR data source with driver: ESRI Shapefile
## Source: "/Users/andrewfox/Desktop/DFP/danube", layer: "discont"
## with 1 features
## It has 1 fields
## Integer64 fields read as strings:  id
```

```
plot(danube)
```

```
danube.sf <- st_read("/Users/andrewfox/Desktop/DFP/danube", layer = "discont")
```

```
## Reading layer `discont' from data source `/Users/andrewfox/Desktop/DFP/danube' using driver `ESRI Sha
## Simple feature collection with 1 feature and 1 field
## Geometry type: LINESTRING
## Dimension:     XY
## Bounding box:  xmin: 18.68919 ymin: 45.91069 xmax: 19.10948 ymax: 47.81571
## CRS:           NA
```

```
ggplot()+geom_sf(data=danube.sf)
```



Now I load the elevation data, joining it with the luminosity data. Since the luminosity point geometries don't cover the entire country, I use the st_nearest_feature option to match each luminosity point with the closest elevation measure. While there is some difference in elevation on either side of the Danube, it's note extreme. Including it in the regressions will allow us to control for this difference.

```
hungary_elev_stars <- read_stars("~/Desktop/DFP/tif files/Hungary_elevation.tif")
plot(hungary_elev_stars)
```


```
hungary.elev.sf <- st_as_sf(hungary_elev_stars, as_points = TRUE, merge = FALSE, long = TRUE, crs = 432
hungary.sf <- st_join(hungary.sf, hungary.elev.sf, join = st_nearest_feature)
```

```
## although coordinates are longitude/latitude, st_nearest_points assumes that they are planar
```

I repeat the same procedure with population data, and soil quality data. I then code

```
hungary_pop_stars <- read_stars("~/Desktop/DFP/hungary.pop.tif")
hungary.pop.sf <- st_as_sf(hungary_pop_stars, as_points = TRUE, merge = FALSE, long = TRUE, crs = 4326,
hungary.sf <- st_join(hungary.sf, hungary.pop.sf, join = st_nearest_feature)
```

```
## although coordinates are longitude/latitude, st_nearest_points assumes that they are planar
```

```
hungary_soil_stars <- read_stars("~/Desktop/DFP/soil.tif")
hungary.soil.sf <- st_as_sf(hungary_soil_stars, as_points = TRUE, merge = FALSE, long = TRUE, crs = 432
hungary.sf <- st_join(hungary.sf, hungary.soil.sf, join = st_nearest_feature)
```

```
## although coordinates are longitude/latitude, st_nearest_points assumes that they are planar
```

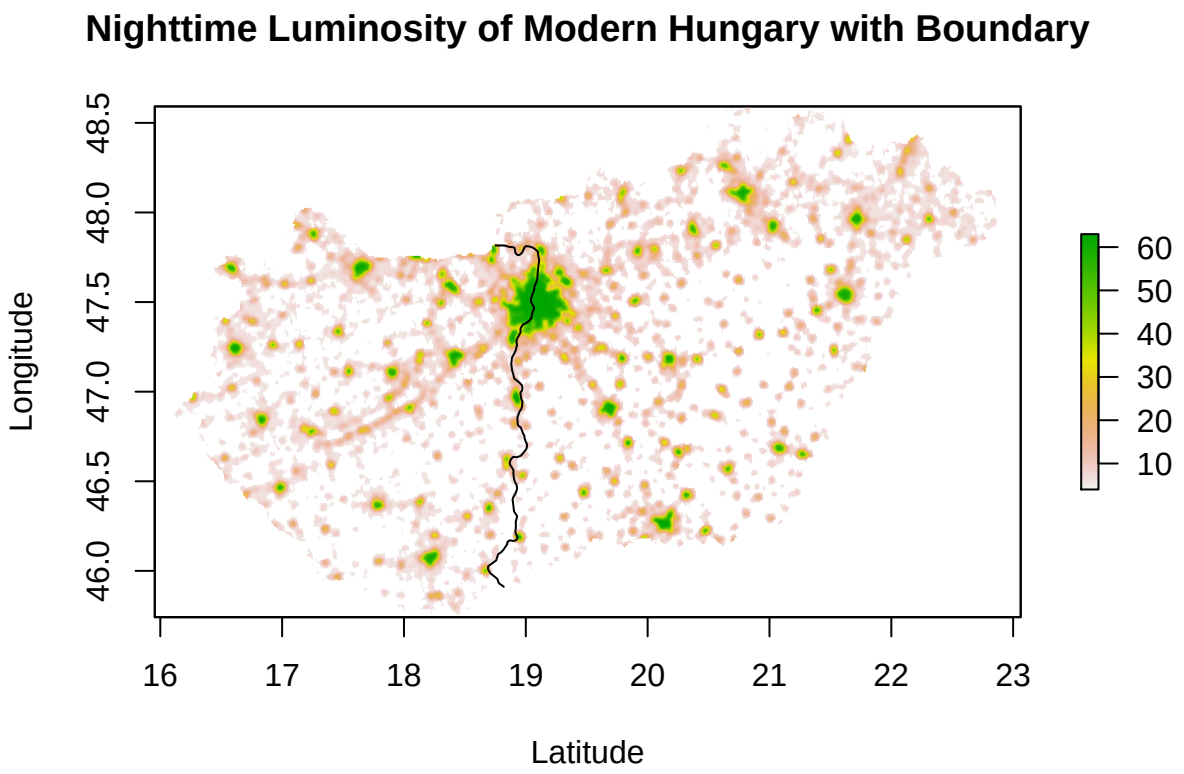
Just coding the soil variable as a proper factor variable...

```
hungary.sf$soil.tf.f <- factor(hungary.sf$soil.tif)
is.factor(hungary.sf$soil.tf.f)
```

```
## [1] TRUE
```

I'll make a quick plot to make sure the boundary looks right.

```
plot(hungary, main = "Nighttime Luminosity of Modern Hungary with Boundary", xlab = "Latitude", ylab =
plot(danube, add = TRUE)
```

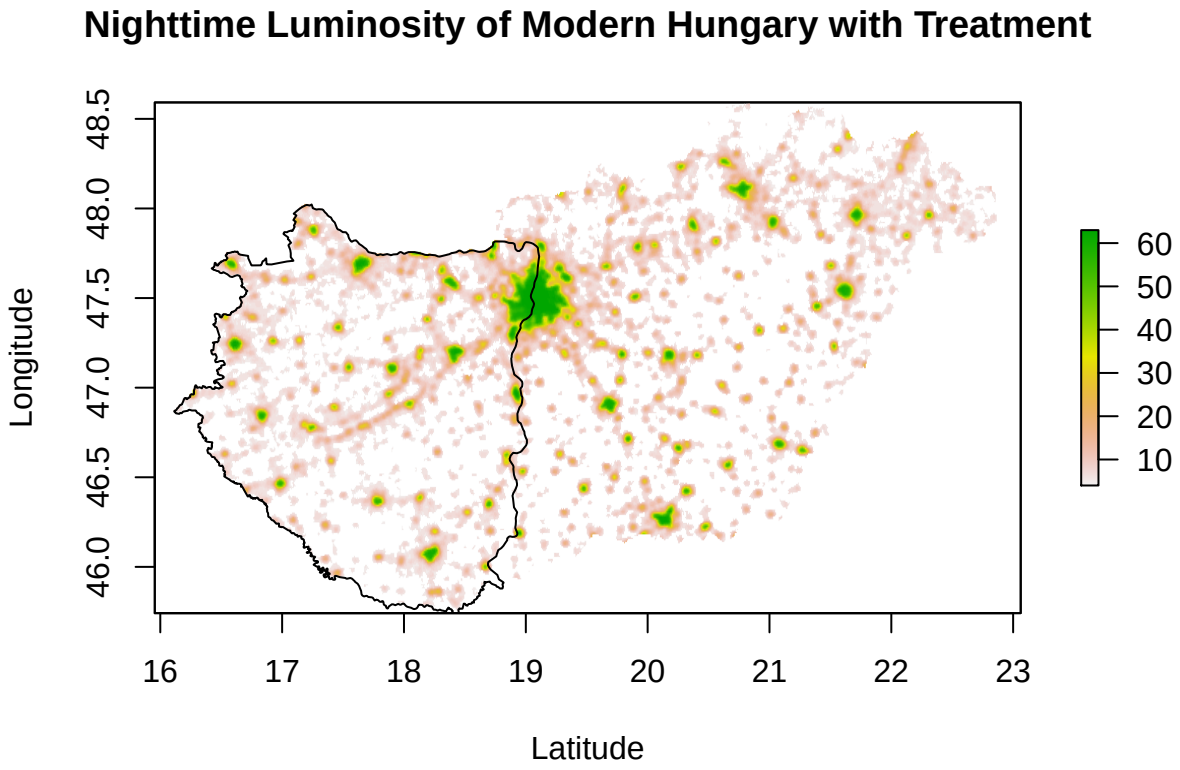


Looks good! Now to import and plot the treatment polygon

```
treat <- readOGR(dsn="/Users/andrewfox/Desktop/DFP/treatment", layer="treat")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "/Users/andrewfox/Desktop/DFP/treatment", layer: "treat"
## with 1 features
## It has 1 fields
## Integer64 fields read as strings:  id
```

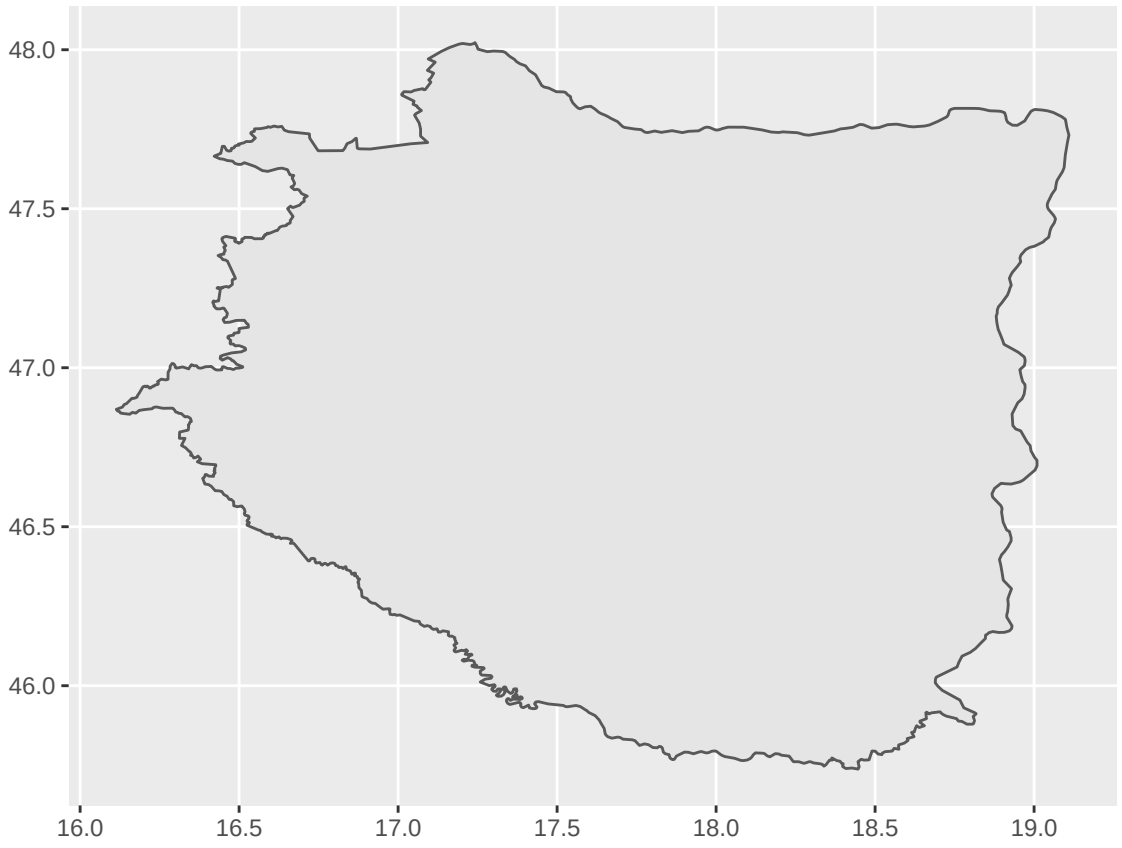
```
plot(hungary, main = "Nighttime Luminosity of Modern Hungary with Treatment", xlab = "Latitude", ylab =
plot(treat, add = TRUE)
```




```
treat.sf <- st_read("/Users/andrewfox/Desktop/DFP/treatment", layer = "treat")
```

```
## Reading layer `treat' from data source `/Users/andrewfox/Desktop/DFP/treatment' using driver `ESRI S
## Simple feature collection with 1 feature and 1 field
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 16.11385 ymin: 45.73784 xmax: 19.10948 ymax: 48.02247
## CRS:           NA
```

```
ggplot()+geom_sf(data=treat.sf)
```



Now to make sure the coordinate systems are aligned for all thre .sf datasets

```
st_crs(treat.sf) = 4326
st_crs(hungary.sf) = 4326
st_crs(danube.sf) = 4326
```

Assigning the treatment area

```
hungary.sf$treated <- assign_treated(hungary.sf, treat.sf, id = "geometry")
```

```
## although coordinates are longitude/latitude, st_intersection assumes that they are planar
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

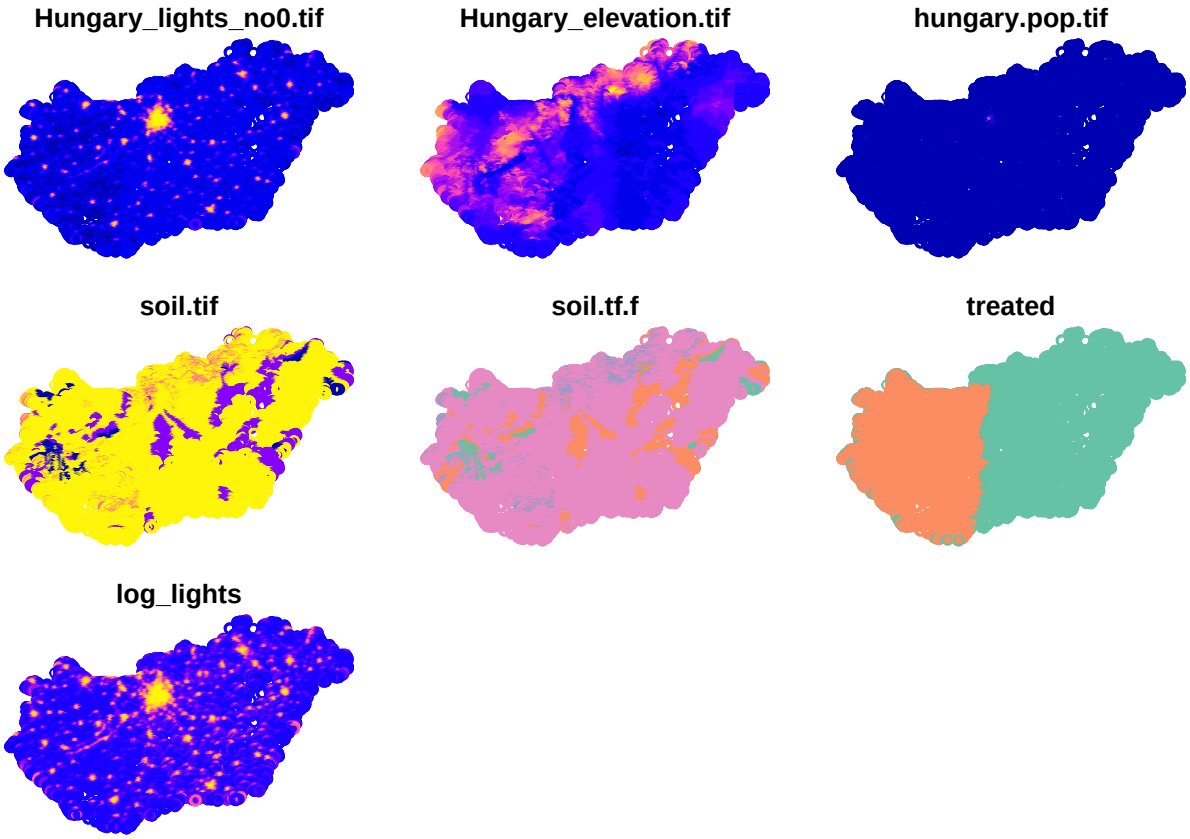
```
# a warning appears, but for this small area, it shouldn't be a problem
summary(hungary.sf$treated)
```

```
##     0     1
## 42093 32892
```

```
hungary.sf$log_lights <- log(hungary.sf$Hungary_lights_no0.tif)
```

Finally here's a plot of all the variables over Hungary

```
plot(hungary.sf)
```



first simple pooled binary variable regressions. From the estimated coefficient on the treatment area we can see that on average light levels are a bit lower in the transdanubian side of Hungary.

```
lm1 <- lm(log_lights ~ treated, data = hungary.sf)
lm2 <- lm(log_lights ~ treated + Hungary_elevation.tif + hungary.pop.tif + soil.tf.f , data = hungary.s
lm1
```

```
##
## Call:
## lm(formula = log_lights ~ treated, data = hungary.sf)
##
## Coefficients:
## (Intercept)     treated1
##     2.29825     -0.07139
```

```
lm2
```

```
##
## Call:
## lm(formula = log_lights ~ treated + Hungary_elevation.tif + hungary.pop.tif +
##     soil.tf.f, data = hungary.sf)
##
## Coefficients:
##           (Intercept)               treated1  Hungary_elevation.tif
##             2.1094420             -0.0313317             -0.0012742
##       hungary.pop.tif             soil.tf.f2             soil.tf.f3
##             0.0002863              0.1234426              0.1665602
```

```
##            soil.tf.f4
##             0.1556673
```

Now I create a new variable measuring the distance to the cutoff.

```
hungary.sf$dist2cutoff <- as.numeric(sf::st_distance(hungary.sf, danube.sf))
summary(hungary.sf$dist2cutoff)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##      2.43  42088.57  92555.17  99646.37 151698.12 282566.73
```

Let's do a whole buncha regressions and display them below.

```
lm4 <- lm(log_lights ~ treated + Hungary_elevation.tif + hungary.pop.tif, data = hungary.sf[hungary.sf$
lm5 <- lm(log_lights ~ treated + Hungary_elevation.tif + hungary.pop.tif + soil.tf.f, data = hungary.sf
lm6 <- lm(log_lights ~ treated + Hungary_elevation.tif + hungary.pop.tif + soil.tf.f, data = hungary.sf
lm7 <- lm(log_lights ~ treated + Hungary_elevation.tif + hungary.pop.tif + soil.tf.f, data = hungary.sf
lm8 <- lm(log_lights ~ treated + Hungary_elevation.tif + hungary.pop.tif + soil.tf.f, data = hungary.sf

lm4
```

```
##
## Call:
## lm(formula = log_lights ~ treated + Hungary_elevation.tif + hungary.pop.tif,
##     data = hungary.sf[hungary.sf$dist2cutoff < 5000, ])
##
## Coefficients:
##           (Intercept)               treated1  Hungary_elevation.tif
##             2.8406073              0.1213826             -0.0030472
##       hungary.pop.tif
##             0.0001388
```

```
lm5
```

```
##
## Call:
## lm(formula = log_lights ~ treated + Hungary_elevation.tif + hungary.pop.tif +
##     soil.tf.f, data = hungary.sf[hungary.sf$dist2cutoff < 10000,
##     ])
##
## Coefficients:
##           (Intercept)               treated1  Hungary_elevation.tif
##             2.1139662              0.0537652              0.0009374
##       hungary.pop.tif             soil.tf.f2             soil.tf.f3
##             0.0001660              0.7106637              0.2649624
##            soil.tf.f4
##             0.5000241
```

```
lm6
```

```
##
## Call:
## lm(formula = log_lights ~ treated + Hungary_elevation.tif + hungary.pop.tif +
##     soil.tf.f, data = hungary.sf[hungary.sf$dist2cutoff < 3000,
##     ])
##
## Coefficients:
##           (Intercept)               treated1  Hungary_elevation.tif
##             2.8400280              0.1081427             -0.0066247
```

```
##       hungary.pop.tif             soil.tf.f2             soil.tf.f3
##             0.0001136              0.0116739              0.0830360
##            soil.tf.f4
##             0.1639961
```

```
lm7
```

```
##
## Call:
## lm(formula = log_lights ~ treated + Hungary_elevation.tif + hungary.pop.tif +
##     soil.tf.f, data = hungary.sf[hungary.sf$dist2cutoff < 5000,
##     ])
##
## Coefficients:
##           (Intercept)               treated1  Hungary_elevation.tif
##             2.4211065              0.1127654             -0.0017372
##       hungary.pop.tif             soil.tf.f2             soil.tf.f3
##             0.0001371              0.1229078              0.1886210
##            soil.tf.f4
##             0.3974020
```

```
lm8
```

```
##
## Call:
## lm(formula = log_lights ~ treated + Hungary_elevation.tif + hungary.pop.tif +
##     soil.tf.f, data = hungary.sf[hungary.sf$dist2cutoff < 10000,
##     ])
##
## Coefficients:
##           (Intercept)               treated1  Hungary_elevation.tif
##             2.1139662              0.0537652              0.0009374
##       hungary.pop.tif             soil.tf.f2             soil.tf.f3
##             0.0001660              0.7106637              0.2649624
##            soil.tf.f4
##             0.5000241
```

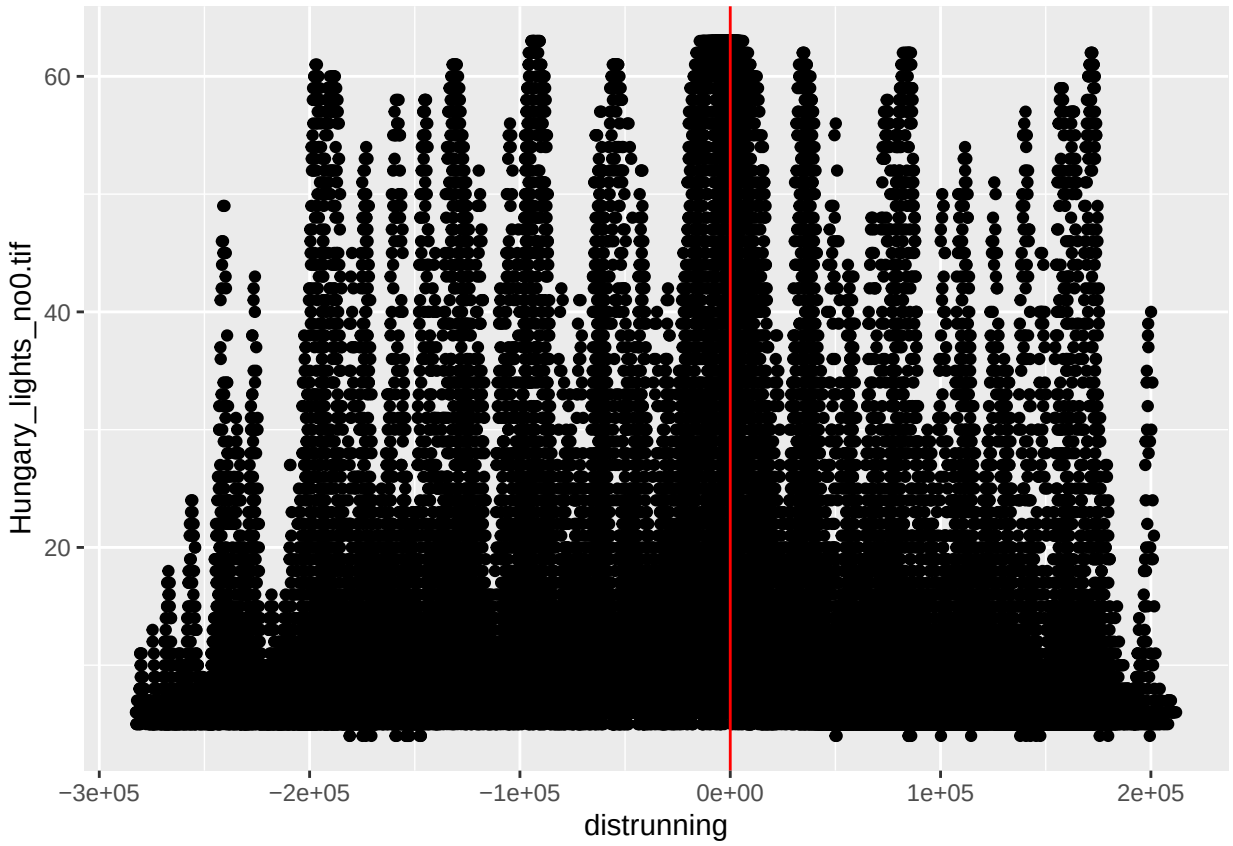
A basic linear regression on a binary variable using the treatment area shows that on average there is lower luminosity on the Roman side of the Danube, while linear regressions with a buffer around the Danube of 3, 5, and 10 km show the effect of being on the Roman side is positive and significant. The linear regression results are in table 2. Elevation and population have small but significant effects.

```
lm3 <- lm(Hungary_lights_no0.tif ~ treated, data = hungary.sf[hungary.sf$dist2cutoff < 15000, ])
coef(summary(lm3))[, "Std. Error"]
```

```
## (Intercept)    treated1
##   0.3164534   0.4379951
```

To prepare for the true spatial RDD, I'll recenter at zero

```
hungary.sf$distrunning <- hungary.sf$dist2cutoff
hungary.sf$distrunning[hungary.sf$treated == 0] <- -1 * hungary.sf$distrunning[hungary.sf$treated == 0]
ggplot(data = hungary.sf, aes(x = distrunning, y = Hungary_lights_no0.tif)) + geom_point() + geom_vline
```



Now some "naive" RDD with local linear estimation

```
covs1 <- cbind(hungary.sf$Hungary_elevation.tif, hungary.sf$hungary.pop.tif)
covs2 <- cbind(hungary.sf$Hungary_elevation.tif, hungary.sf$hungary.pop.tif, hungary.sf$soil.tf.f)
rd1 <- rdrobust(hungary.sf$log_lights, hungary.sf$distrunning, c = 0)
rd1.1 <- rdrobust(hungary.sf$log_lights, hungary.sf$distrunning, c = 0, p = 2)
rd2 <- rdrobust(hungary.sf$log_lights, hungary.sf$distrunning, covs = hungary.sf$Hungary_elevation.tif,
rd3 <- rdrobust(hungary.sf$log_lights, hungary.sf$distrunning, covs = covs1, c = 0)
rd4 <- rdrobust(hungary.sf$log_lights, hungary.sf$distrunning, covs = covs2, c = 0)
rd5 <- rdrobust(hungary.sf$log_lights, hungary.sf$distrunning, covs = covs1, c = 0, p=2)
rd6 <- rdrobust(hungary.sf$log_lights, hungary.sf$distrunning, covs = covs2, c = 0, p =2)
summary(rd1)
```

```
## Call: rdrobust
##
## Number of Obs.                 74985
## BW type                        mserd
## Kernel                     Triangular
## VCE method                         NN
##
## Number of Obs.                 42093       32892
## Eff. Number of Obs.             3561        3828
## Order est. (p)                     1           1
## Order bias  (q)                    2           2
## BW est. (h)                13579.435   13579.435
## BW bias (b)                28909.696   28909.696
## rho (h/b)                      0.470       0.470
## Unique Obs.                    42093       32892
```

```
##
## ===============================================================================
##         Method     Coef. Std. Err.         z     P>|z|      [ 95% C.I. ]
## ===============================================================================
##   Conventional     0.096     0.039     2.483     0.013     [0.020 , 0.172]
##         Robust         -         -     2.766     0.006     [0.034 , 0.200]
## ===============================================================================
```

```
summary(rd1.1)
```

```
## Call: rdrobust
##
## Number of Obs.                74985
## BW type                       mserd
## Kernel                   Triangular
## VCE method                       NN
##
## Number of Obs.               42093       32892
## Eff. Number of Obs.           5483        5963
## Order est. (p)                   2           2
## Order bias  (q)                  3           3
## BW est. (h)              23758.205   23758.205
## BW bias (b)              52192.749   52192.749
## rho (h/b)                    0.455       0.455
## Unique Obs.                  42093       32892
##
## ===============================================================================
##         Method     Coef. Std. Err.         z     P>|z|      [ 95% C.I. ]
## ===============================================================================
##   Conventional     0.114     0.043     2.661     0.008     [0.030 , 0.197]
##         Robust         -         -     2.995     0.003     [0.045 , 0.217]
## ===============================================================================
```

```
summary(rd2)
```

```
## Call: rdrobust
##
## Number of Obs.                74985
## BW type                       mserd
## Kernel                   Triangular
## VCE method                       NN
##
## Number of Obs.               42093       32892
## Eff. Number of Obs.           3237        3438
## Order est. (p)                   1           1
## Order bias  (q)                  2           2
## BW est. (h)              11940.105   11940.105
## BW bias (b)              26485.922   26485.922
## rho (h/b)                    0.451       0.451
## Unique Obs.                  42093       32892
##
## ===============================================================================
##         Method     Coef. Std. Err.         z     P>|z|      [ 95% C.I. ]
## ===============================================================================
##   Conventional     0.101     0.041     2.493     0.013     [0.022 , 0.181]
```

```
##         Robust         -         -     2.555     0.011     [0.026 , 0.199]
## ================================================================================
```

```
summary(rd3)
```

```
## Call: rdrobust
##
## Number of Obs.                74985
## BW type                       mserd
## Kernel                   Triangular
## VCE method                       NN
##
## Number of Obs.               42093       32892
## Eff. Number of Obs.           3364        3597
## Order est. (p)                   1           1
## Order bias  (q)                  2           2
## BW est. (h)              12591.005   12591.005
## BW bias (b)              29251.829   29251.829
## rho (h/b)                    0.430       0.430
## Unique Obs.                  42093       32892
##
## ================================================================================
##         Method     Coef. Std. Err.         z     P>|z|      [ 95% C.I. ]
## ================================================================================
##   Conventional     0.146     0.036     4.012     0.000     [0.074 , 0.217]
##         Robust         -         -     4.128     0.000     [0.085 , 0.238]
## ================================================================================
```

```
summary(rd4)
```

```
## Call: rdrobust
##
## Number of Obs.                74985
## BW type                       mserd
## Kernel                   Triangular
## VCE method                       NN
##
## Number of Obs.               42093       32892
## Eff. Number of Obs.           3299        3511
## Order est. (p)                   1           1
## Order bias  (q)                  2           2
## BW est. (h)              12236.762   12236.762
## BW bias (b)              27726.531   27726.531
## rho (h/b)                    0.441       0.441
## Unique Obs.                  42093       32892
##
## ================================================================================
##         Method     Coef. Std. Err.         z     P>|z|      [ 95% C.I. ]
## ================================================================================
##   Conventional     0.145     0.037     3.947     0.000     [0.073 , 0.217]
##         Robust         -         -     3.967     0.000     [0.080 , 0.235]
## ================================================================================
```

```
summary(rd5)
```

```
## Call: rdrobust
```

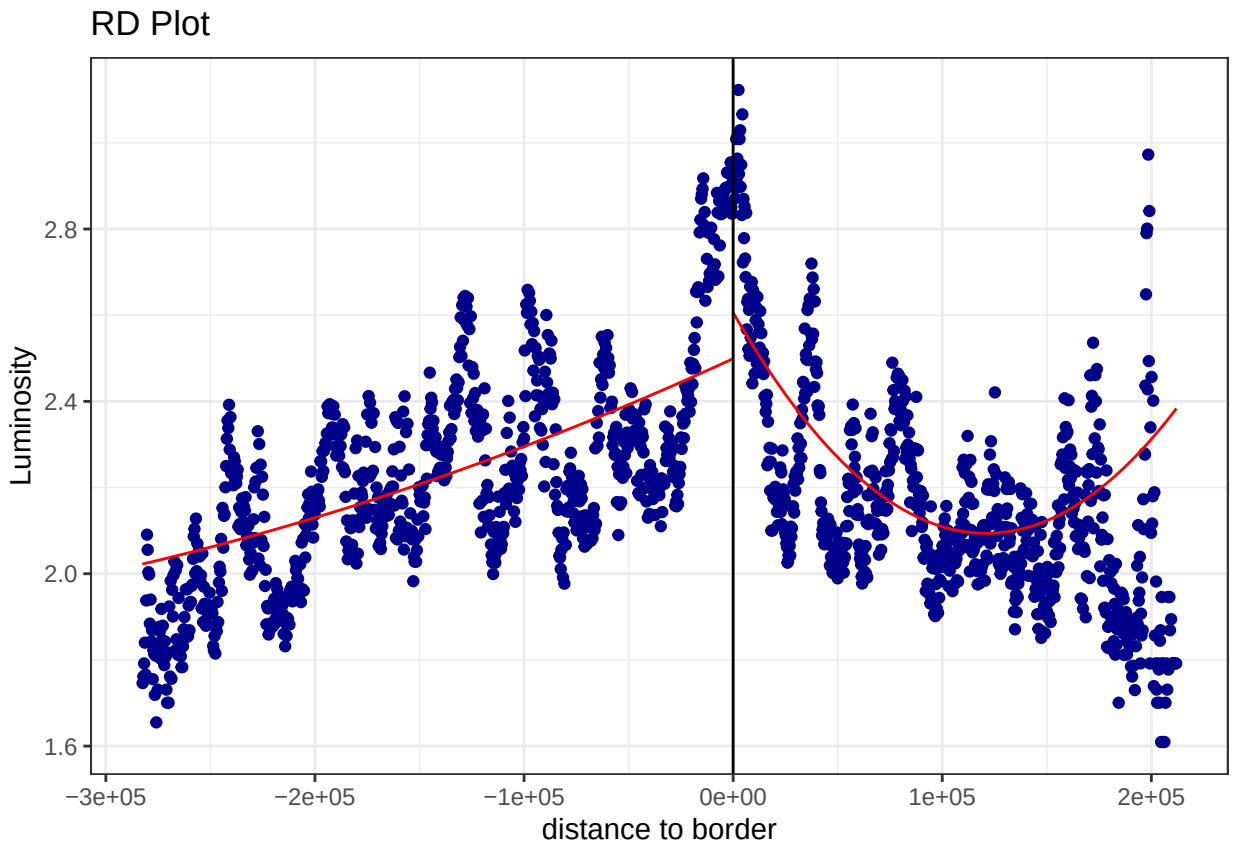
```
##
## Number of Obs.                74985
## BW type                       mserd
## Kernel                   Triangular
## VCE method                       NN
##
## Number of Obs.               42093       32892
## Eff. Number of Obs.           5749        6214
## Order est. (p)                   2           2
## Order bias  (q)                  3           3
## BW est. (h)              25253.257   25253.257
## BW bias (b)              60339.578   60339.578
## rho (h/b)                    0.419       0.419
## Unique Obs.                  42093       32892
##
## ===============================================================================
##         Method     Coef. Std. Err.         z     P>|z|      [ 95% C.I. ]
## ===============================================================================
##   Conventional     0.171     0.038     4.455     0.000     [0.096 , 0.246]
##         Robust         -         -     4.746     0.000     [0.109 , 0.262]
## ===============================================================================
```

```
summary(rd6)
```

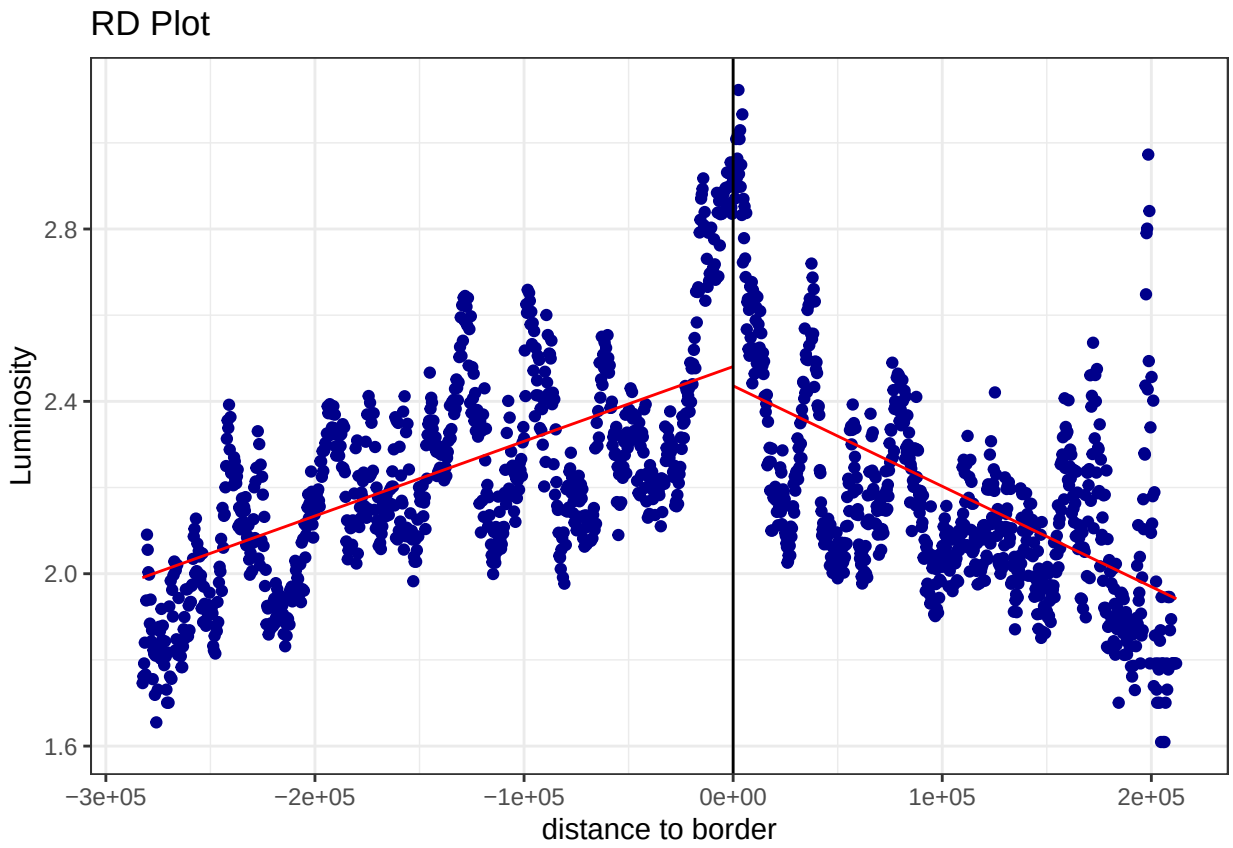
```
## Call: rdrobust
##
## Number of Obs.                74985
## BW type                       mserd
## Kernel                   Triangular
## VCE method                       NN
##
## Number of Obs.               42093       32892
## Eff. Number of Obs.           6545        7069
## Order est. (p)                   2           2
## Order bias  (q)                  3           3
## BW est. (h)              29538.291   29538.291
## BW bias (b)              71288.391   71288.391
## rho (h/b)                    0.414       0.414
## Unique Obs.                  42093       32892
##
## ===============================================================================
##         Method     Coef. Std. Err.         z     P>|z|      [ 95% C.I. ]
## ===============================================================================
##   Conventional     0.189     0.036     5.303     0.000     [0.119 , 0.260]
##         Robust         -         -     5.535     0.000     [0.130 , 0.273]
## ===============================================================================
```

Well, that was cool. You can see that now all the estimated coefficients on the treatment area are positive. Now for some RD plots.

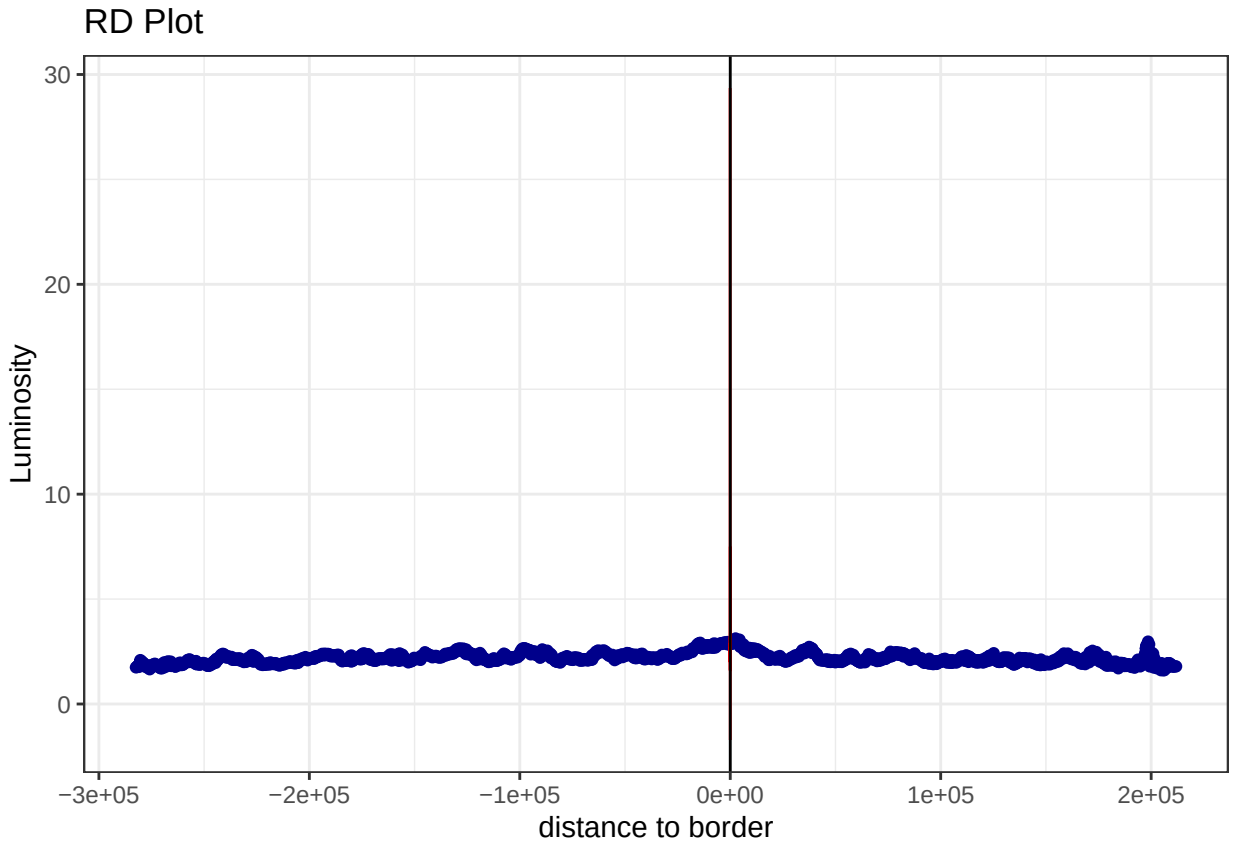
```
rdplot(hungary.sf$log_lights, hungary.sf$distrunning, covs = covs2, c = 0, p =2,
       y.label = "Luminosity", x.label = "distance to border")
```



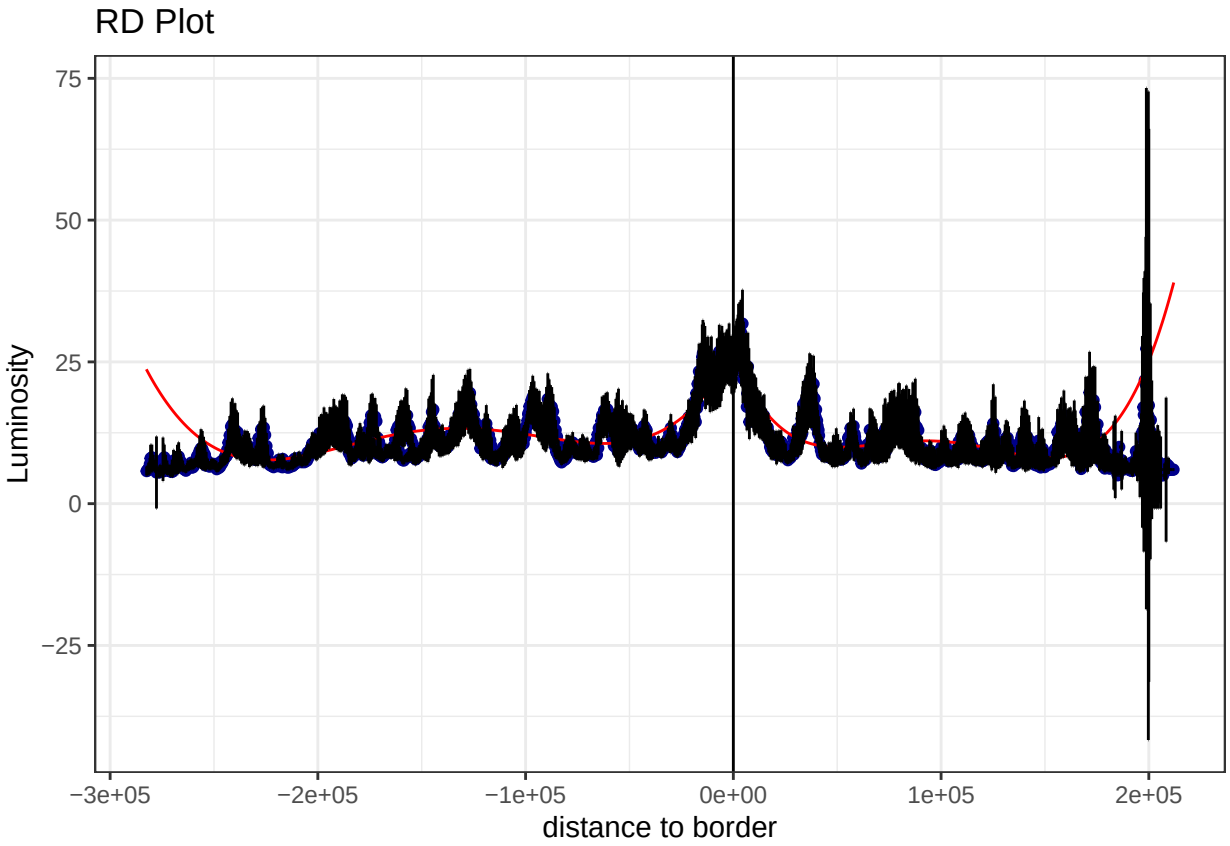

```
rdplot(hungary.sf$log_lights, hungary.sf$distrunning, covs = covs2, c = 0, p =1,
       y.label = "Luminosity", x.label = "distance to border")
```




```
rdplot(hungary.sf$log_lights, hungary.sf$distrunning, covs = covs2, c = 0, p =6, h = 60,
       y.label = "Luminosity", x.label = "distance to border")
```




```
rdplot(hungary.sf$Hungary_lights_no0.tif, hungary.sf$distrunning, c = 0, ci = 95,
       kernel = "triangular", y.label = "Luminosity", x.label = "distance to border")
```



The second plot in particular suggests that linear regression on either side reveals the positive discontinuity at the Danube.

For a more precise look, I discretize the border line I cut earlier, first every 6km, then every 1.

```
borderpoints.sf <- discretise_border(cutoff = danube.sf, n = 50)
```

```
## Starting to create 50 borderpoints from the given set of borderpoints. Approximately every 6 kilometi
```

```
borderpoints.sf$id <- 1:nrow(borderpoints.sf)
borderpoints.sf2 <- discretise_border(cutoff = danube.sf, n = 10)
```

```
## Starting to create 10 borderpoints from the given set of borderpoints. Approximately every 28 kilome
```

```
borderpoints.sf2$id <- 1:nrow(borderpoints.sf2)
```

Finally by looping over boundary points with spatialrd(), with 50 sections, I'll produce non-parametric estimates of the treatment effect

```
bdd1 <- spatialrd(y = "log_lights", data = hungary.sf, cutoff.points = borderpoints.sf,
treated = "treated", minobs = 10, spatial.object = T)
```

```
## We have 74985 observations of which 32892 are treated observations.
## We are iterating over 50 Boundarypoints.
## The dependent variable is log_lights .
```

```
## Warning in log(fhatl): NaNs produced
```

```
## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced
```

```
## Warning in log(fhatl): NaNs produced
```

```
## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced

## Warning in log(fhatl): NaNs produced

## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced

## Warning in log(fhatl): NaNs produced

## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced

## Warning in log(fhatr): NaNs produced

## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced

## Warning in log(fhatr): NaNs produced

## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced

## Warning in log(fhatr): NaNs produced

## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced

## Warning in log(fhatr): NaNs produced

## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced

## Warning in log(fhatr): NaNs produced

## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced

## Warning in log(fhatr): NaNs produced

## Warning in log(fhatr): NaNs produced
```

```r
knitr::kable(bdd1)
```

Point	Estimate	pvalC	pvalR	Ntr	Nco	bw	CI_Conv_l	CI_Conv_u	CI_Rob_l	CI_Rob_u	geometry
1	0.63	0.00	0.00	355	823	20.2	0.37	0.90	0.32	0.87	POINT (18.78747 47.81561)
2	-0.26	0.00	0.00	380	690	17.8	-0.38	-0.14	-0.43	-0.15	POINT (18.85868 47.80836)
3	-0.40	0.00	0.00	149	193	8.9	-0.54	-0.27	-0.59	-0.31	POINT (18.91163 47.78733)
4	-0.99	0.00	0.00	129	296	9.2	-1.17	-0.82	-1.22	-0.87	POINT (18.95732 47.7683)
5	-0.48	0.00	0.00	228	137	8.6	-0.61	-0.34	-0.67	-0.39	POINT (18.99494 47.80897)
6	-0.71	0.00	0.00	327	152	9.7	-0.90	-0.51	-0.96	-0.57	POINT (19.06347 47.80122)
7	-1.10	0.00	0.00	595	301	13.2	-1.24	-0.96	-1.28	-1.01	POINT (19.10172 47.76339)
8	-0.36	0.00	0.00	1076	649	18.3	-0.50	-0.22	-0.54	-0.26	POINT (19.10615 47.71501)
9	-0.19	0.00	0.00	1554	1083	22.5	-0.30	-0.09	-0.34	-0.08	POINT (19.09792 47.66632)
10	-0.18	0.00	0.00	2220	1700	27.4	-0.25	-0.11	-0.27	-0.09	POINT (19.0898 47.61775)
11	0.06	0.00	0.04	321	327	11.0	0.03	0.09	0.00	0.08	POINT (19.06849 47.57119)
12	0.06	0.00	0.00	248	326	10.3	0.05	0.07	0.02	0.07	POINT (19.04572 47.52489)

Point	Estimate	pvalC	pvalR	Ntr	Nco	bw	CI_Conv_C	CI_Conv_C	CI_Rob_C	CI_Rob_C	geometry
13	0.02	0.00	0.00	337	282	10.7	0.01	0.03	0.02	0.05	POINT (19.06122 47.4793)
14	0.00	0.71	0.00	183	149	7.9	-0.02	0.01	-0.04	-0.01	POINT (19.05091 47.43269)
15	0.04	0.04	0.99	101	81	5.8	0.00	0.08	-0.04	0.04	POINT (19.01933 47.39108)
16	-0.29	0.00	0.00	124	175	7.4	-0.42	-0.17	-0.47	-0.21	POINT (18.96433 47.36088)
17	0.79	0.00	0.00	395	446	12.6	0.65	0.94	0.67	0.98	POINT (18.94467 47.31517)
18	0.92	0.00	0.00	655	707	16.7	0.76	1.09	0.80	1.13	POINT (18.92553 47.26944)
19	0.10	0.13	0.21	1249	1704	27.0	-0.03	0.23	-0.06	0.25	POINT (18.91156 47.22204)
20	-0.70	0.00	0.00	213	255	9.8	-0.90	-0.51	-0.95	-0.56	POINT (18.88413 47.17686)
21	0.34	0.00	0.00	188	185	9.1	0.23	0.45	0.19	0.42	POINT (18.8857 47.12778)
22	0.32	0.00	0.00	537	739	18.8	0.20	0.43	0.23	0.46	POINT (18.90327 47.07974)
23	0.60	0.00	0.00	324	271	12.4	0.44	0.75	0.48	0.80	POINT (18.95398 47.04586)
24	1.34	0.00	0.00	202	262	10.9	1.08	1.60	1.13	1.66	POINT (18.96394 47.00164)
25	1.53	0.00	0.00	173	327	12.0	1.35	1.70	1.39	1.75	POINT (18.96543 46.95474)
26	0.77	0.00	0.00	305	523	17.7	0.55	0.98	0.59	1.04	POINT (18.96416 46.90658)
27	0.33	0.00	0.00	589	939	27.6	0.21	0.44	0.23	0.49	POINT (18.93426 46.86174)
28	0.22	0.00	0.00	1185	1456	38.2	0.14	0.31	0.12	0.33	POINT (18.93735 46.81313)
29	-0.55	0.00	0.00	183	229	14.8	-0.71	-0.40	-0.75	-0.43	POINT (18.97436 46.77275)
30	-0.42	0.00	0.00	233	233	16.3	-0.52	-0.32	-0.56	-0.35	POINT (18.99715 46.72607)
31	-0.25	0.00	0.00	547	435	22.9	-0.32	-0.17	-0.35	-0.18	POINT (19.00342 46.67867)
32	0.14	0.04	0.01	257	302	16.3	0.01	0.28	0.04	0.32	POINT (18.95565 46.6412)
33	0.08	0.17	0.30	1075	1723	41.0	-0.03	0.18	-0.06	0.20	POINT (18.88873 46.63074)
34	0.57	0.00	0.00	317	530	20.6	0.44	0.70	0.47	0.75	POINT (18.87621 46.58653)
35	-0.02	0.77	0.74	751	1101	34.1	-0.14	0.11	-0.18	0.12	POINT (18.89852 46.54091)
36	-0.07	0.21	0.25	744	1129	34.7	-0.18	0.04	-0.21	0.06	POINT (18.9128 46.49202)
37	0.15	0.01	0.06	131	119	11.2	0.03	0.26	-0.01	0.22	POINT (18.92271 46.44589)
38	0.12	0.01	0.46	364	613	21.0	0.03	0.22	-0.08	0.17	POINT (18.89266 46.40044)

Point	Estimate	pvalC	pvalR	Ntr	Nco	bw	CI_Conv_l	CI_Conv_r	CI_Rob_l	CI_Rob_r	geometry
39	0.40	0.00	0.00	395	667	22.6	0.26	0.53	0.30	0.58	POINT (18.90007 46.35082)
40	0.20	0.02	0.05	755	1019	31.5	0.03	0.37	0.00	0.42	POINT (18.92817 46.30588)
41	-0.09	0.30	0.29	866	1153	34.3	-0.26	0.08	-0.31	0.09	POINT (18.91873 46.25676)
42	-0.41	0.01	0.00	353	261	19.4	-0.72	-0.11	-0.79	-0.17	POINT (18.91919 46.20736)
43	-0.05	0.72	0.40	379	296	20.7	-0.33	0.23	-0.42	0.17	POINT (18.90582 46.16784)
44	-0.01	0.96	0.74	300	369	19.1	-0.33	0.32	-0.40	0.28	POINT (18.8472 46.15362)
45	0.36	0.00	0.00	311	487	21.3	0.24	0.48	0.18	0.43	POINT (18.8087 46.11183)
46	-0.20	0.00	0.00	815	3091	50.8	-0.30	-0.10	-0.34	-0.10	POINT (18.76395 46.0747)
47	-0.05	0.32	0.25	1038	4218	62.4	-0.14	0.05	-0.18	0.05	POINT (18.71758 46.03925)
48	-0.01	0.80	0.46	1064	4404	67.2	-0.08	0.06	-0.11	0.05	POINT (18.69682 45.99937)
49	-0.04	0.37	0.28	716	3215	56.1	-0.14	0.05	-0.17	0.05	POINT (18.74909 45.96482)
50	0.10	0.01	0.02	1500	4673	80.7	0.02	0.17	0.01	0.19	POINT (18.78997 45.92562)

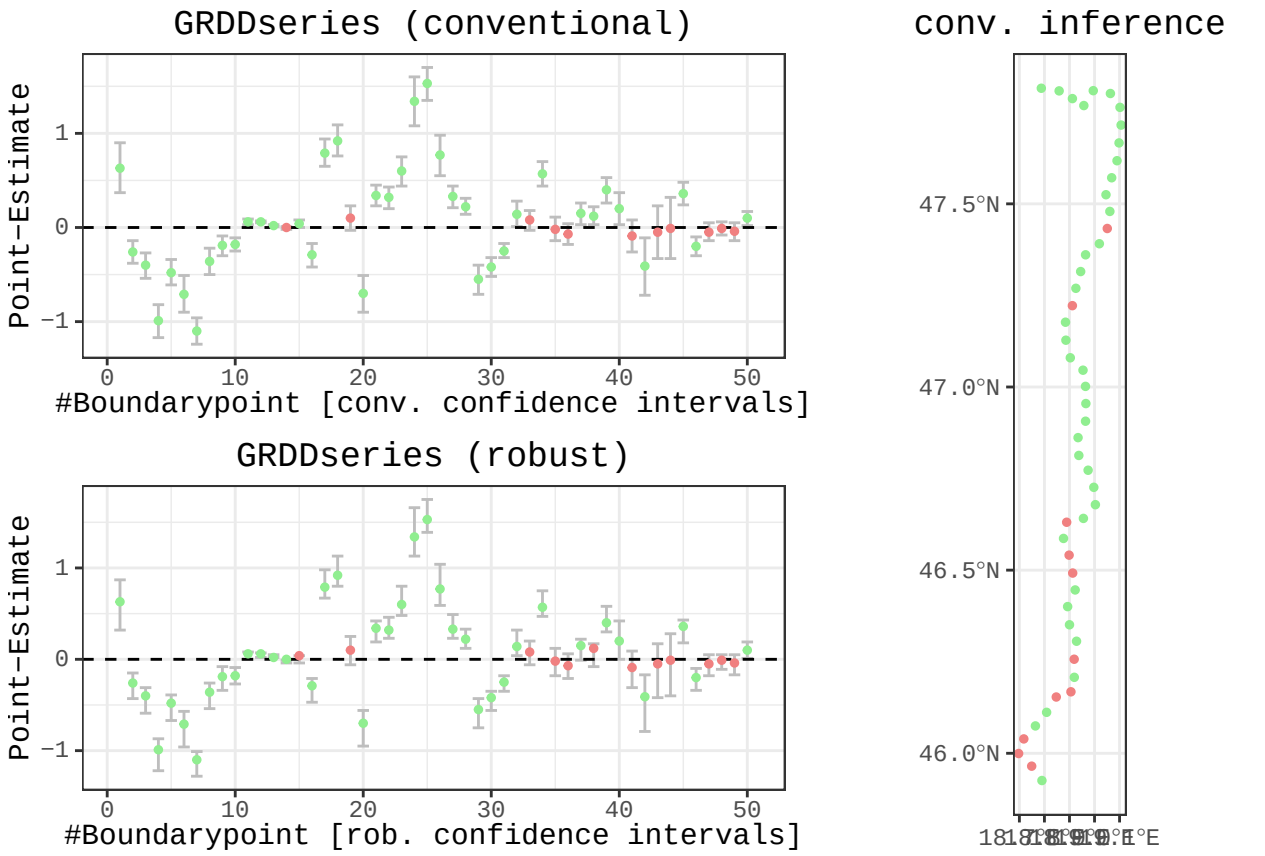
```
bdd2 <- spatialrd(y = "log_lights", data = hungary.sf, cutoff.points = borderpoints.sf,
                  treated = "treated", minobs = 10, covs = covs1, spatial.object = T)
```

```
## We have 74985 observations of which 32892 are treated observations.
## We are iterating over 50 Boundarypoints.
## The dependent variable is log_lights .
## Warning in log(fhatl): NaNs produced
## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced
## Warning in log(fhatl): NaNs produced
## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced
## Warning in log(fhatl): NaNs produced
## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced
## Warning in log(fhatl): NaNs produced
## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced
## Warning in log(fhatr): NaNs produced
## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced
## Warning in log(fhatr): NaNs produced
## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced
## Warning in log(fhatr): NaNs produced
## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced
```

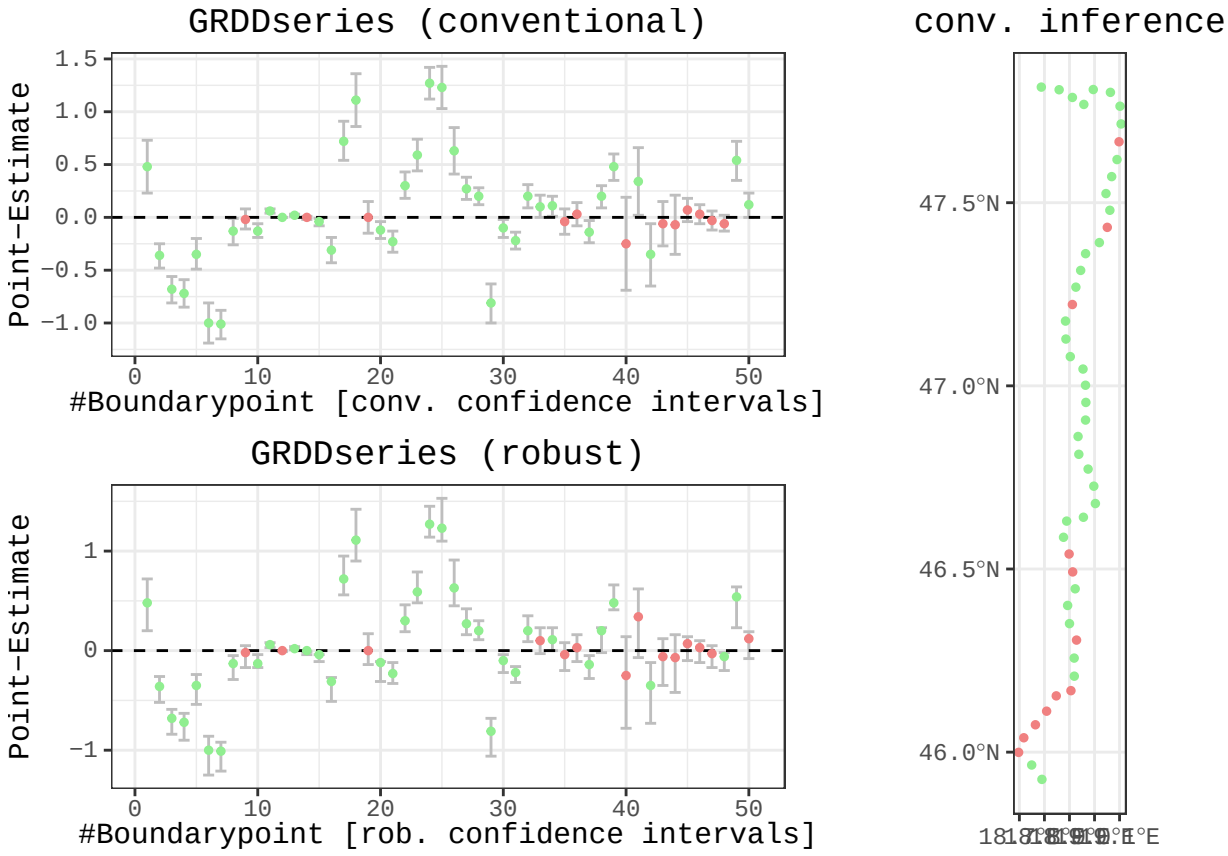
```
## Warning in log(fhatr): NaNs produced
## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced
## Warning in log(fhatr): NaNs produced
## Warning in sqrt((1/(rn * bw)) * (24/5) * ((1/fhatr) + (1/fhatl))): NaNs produced
## Warning in log(fhatr): NaNs produced

## Warning in log(fhatr): NaNs produced
```

my little laptop had trouble running the third regression, but it isn't used in the paper, so I'll present the RD plots for the first two

```
plotspatialrd(bdd1, map = T)
```




```
plotspatialrd(bdd2, map = T)
```



Average treatment effects are calculated below

```
mean(bdd1$Estimate)
```

```
## [1] 0.0472
```

```
mean(bdd1$pvalR)
```

```
## [1] 0.111
```

```
mean(bdd2$Estimate)
```

```
## [1] 0.0374
```

```
mean(bdd2$pvalR)
```

```
## [1] 0.1244
```

Poisson regressions

```
m1 <- glm(Hungary_lights_no0.tif ~ treated + Hungary_elevation.tif + hungary.pop.tif + soil.tf.f, family

m2 <- glm(Hungary_lights_no0.tif ~ treated + Hungary_elevation.tif
          + hungary.pop.tif + soil.tf.f, family="poisson", data=hungary.sf[hungary.sf$dist2cutoff < 300

m3 <- glm(Hungary_lights_no0.tif ~ treated + Hungary_elevation.tif
          + hungary.pop.tif + soil.tf.f, family="poisson", data=hungary.sf[hungary.sf$dist2cutoff < 500

m4 <- glm(Hungary_lights_no0.tif ~ treated + Hungary_elevation.tif
          + hungary.pop.tif + soil.tf.f, family="poisson", data=hungary.sf[hungary.sf$dist2cutoff < 100
```

The actual estimates aren't very informative, so I'll display the marginal estimates instead. Once again, we see a negative marginal effect for the entirety of Hungary. However, restricting the regressions to bands of 3, 5, and 10 km around the cutoff leads to positive and statistically significant coefficients on the treatment variable. Marginal increases are calculated at 2?4 luminosity counts along the right bank within the Roman area, from the 1?63 count measure produced by the NOAA.

```
margins(m1)
```

```
## Average marginal effects
```

```
## glm(formula = Hungary_lights_no0.tif ~ treated + Hungary_elevation.tif +     hungary.pop.tif + soil.t
```

```
##  Hungary_elevation.tif hungary.pop.tif treated1 soil.tf.f2 soil.tf.f3
##               -0.01736        0.001763 -0.03487        1.9      2.225
##  soil.tf.f4
##       2.412
```

```
margins(m2)
```

```
## Average marginal effects
```

```
## glm(formula = Hungary_lights_no0.tif ~ treated + Hungary_elevation.tif +     hungary.pop.tif + soil.t
```

```
##  Hungary_elevation.tif hungary.pop.tif treated1 soil.tf.f2 soil.tf.f3
##                -0.2222        0.001708     3.58     0.3676      4.647
##  soil.tf.f4
##       7.723
```

```
margins(m3)
```

```
## Average marginal effects
```

```
## glm(formula = Hungary_lights_no0.tif ~ treated + Hungary_elevation.tif +     hungary.pop.tif + soil.t
```

```
##  Hungary_elevation.tif hungary.pop.tif treated1 soil.tf.f2 soil.tf.f3
##               -0.05302        0.001971    3.782      3.342      6.925
##  soil.tf.f4
##       12.22
```

```
margins(m4)
```

```
## Average marginal effects
```

```
## glm(formula = Hungary_lights_no0.tif ~ treated + Hungary_elevation.tif +     hungary.pop.tif + soil.t
```

```
##  Hungary_elevation.tif hungary.pop.tif treated1 soil.tf.f2 soil.tf.f3
##                0.02154         0.00213    1.973      17.14      7.272
##  soil.tf.f4
##       13.06
```