What are the Best and Worst Neighborhoods in Portland?

Ian Davis

June 12, 2018

Objective

The objective of this analysis is to create an index of the best and worst neighborhoods in the Portland Metro area. The following factors were used :

Real estate factors

- The percentage of distressed properties within each neighborhood.
- Percentage of condo sales

Amenity factors

- Miles of bike lanes
- Total acreage of parks and open spaces

I chose these 4 variables because I enjoy biking around Portland and spending time in parks and open spaces. I chose percentage of distressed properties because it speaks to the general desirability of a neighbor and the make up of potential neighbors. I chose percentage of condo sales because I prefer living in high-density neighborhoods.

Data sets used

The following data sets were used for this analysis:

- Neighborhood boundaries and names (RLIS/BOUNDARY/nbo_hood.shp)
- Parks and open spaces (RLIS/BOUNDARY/park.dst.shp)
- Bikes lanes (RLIS/TRANSIT/bike_routes.shp)
- Real estate infomation (provided with assignment)

Analytic Method (General)

- Count miles of bike lanes within each neighborhood
- Count acres of parks and open spaces within each neighborhood
- Add real estate information for each neighborhood
- Use an unweighted classification for produce a index of neighborhood scores

Analytic Method (Detail)

Count miles of bike lanes within each neighborhood

1. Use the Summarize Within function, in ArcGIS Pro, add up the length of bike routes for each neighborhood. This adds these fields to the neighborhood attribute table.

Count acres of parks and open spaces within each neighborhood

1. Use the Summarize Within function with feature class created in previous step and park acreage.

Add real estate information for each neighborhood

1. Save the provided Excel workbook as .xls file
2. In an empty ArcMap document (this only works in ArcMap), define the Coordinate Reference System as WGS84. This is so when we Display the XY data from the Excel workbook it CRS will be defined.
3. Add 2011 real estate data worksheet to ArcMap document
4. Right-click and select Display the XY Data. We are warned that we will not be able to perform any queries on the data until it is exported as a feature class. Click OK.
5. Right-click on resulting point feature and export data as Feature Class to my project geodatabase.
6. Import real estate feature class.

Aggregate features

1. Perform Spatial Join to get connect neighborhood, park, bike and real estate data into one table. *The data processing above results in a flat data table that is over 250MB. The best way to handle the task of cleaning this up is to export it as a .csv file and manipulate it with R.*

```
library(sf)
## Linking to GEOS 3.6.1, GDAL 2.2.3, proj.4 4.9.3
library(tmap)
library(dplyr)
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

- Import and test GIANT shape file exported from Arc

```
main_df <- st_read("data/REAL/real_estate_w_parks_bikes_hood.shp")
```

```
## Reading layer `real_estate_w_parks_bikes_hood' from data source `C:\Users\aloosefish\Documents\GeoDa
## Simple feature collection with 21918 features and 31 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 7435745 ymin: 449507.3 xmax: 7898712 ymax: 835546.5
## epsg (SRID):    NA
## proj4string:    +proj=lcc +lat_1=44.33333333333334 +lat_2=46 +lat_0=43.66666666666666 +lon_0=-120.5 +
```

```
names(main_df)
```

```
##  [1] "Join_Count" "TARGET_FID" "JOIN_FID"   "NAME"       "AREA"
##  [6] "SUM_Area_A" "SUM_AREA"   "SUM_Length" "SUM_LENG_1" "PropType"
## [11] "Status"     "DOM"        "CDOM"       "County"     "Zip"
## [16] "Sub_Prop"   "Price_Sale" "Date_COE"   "Latitude"   "Longitude"
## [21] "Bank_Owned" "Short_Sale" "Street_Num" "Street_Dir" "Street"
## [26] "Street_Typ" "Street_Qua" "Unit_No"    "City"       "Shape_Leng"
## [31] "Shape_Area" "geometry"
```

```r
total_rows <- nrow(main_df)
```

- Select just columns needed for analysis.
- Create new columns for aggregate county-level stats for distressed property and condo sales.

```r
slt <- main_df %>%
  as.data.frame %>%
  select(
    NAME,
    Sub_Prop,
    Bank_Owned,
    AREA,
    Short_Sale,
    bike_miles = SUM_Length,
    park_acres = SUM_Area_A,
    Shape_Leng,
    Shape_Area,
    geometry
  ) %>%
  group_by(NAME) %>%
  mutate(
    stressed = (sum((Bank_Owned == "Y") |
                    Short_Sale == "Y") / total_rows) *100,
    condo = (sum(Sub_Prop == "CONDO") / total_rows
  ) * 100) %>%
  distinct(NAME, .keep_all = TRUE)
```

- Dispose of imported file.

```r
remove(main_df)
```

- Drop unneeded columns.

```r
names(slt)
```

```
##  [1] "NAME"       "Sub_Prop"   "Bank_Owned" "AREA"       "Short_Sale"
##  [6] "bike_miles" "park_acres" "Shape_Leng" "Shape_Area" "geometry"
## [11] "stressed"   "condo"
```

```r
no_need <- c("Bank_Owned", "Short_Sale", "Sub_Prop")
slt <- slt %>%
  select(-no_need)
names(slt)
```

```
## [1] "NAME"       "AREA"       "bike_miles" "park_acres" "Shape_Leng"
## [6] "Shape_Area" "geometry"   "stressed"   "condo"
```

- Add index score expression column.

```r
slt <- slt %>%
  mutate(index_score = sum((.25 * bike_miles), (.25 * park_acres), (.25 * condo)) - (.25 * stressed))
```

- Get best and worst ten neighborhoods.

```r
top_10 <- slt %>%
  select(NAME, index_score) %>%
  group_by(NAME) %>%
  arrange(desc(index_score)) %>%
  top_n(10)
```

```
## Selecting by index_score
```

```r
bottom_10 <- slt %>%
  select(NAME, index_score) %>%
  group_by(NAME) %>%
  arrange(index_score) %>%
  top_n(10)
```

```
## Selecting by index_score
```

- Turn back into sf object. Then simplify the polygons so that they render more quickly.

```r
slt <- st_sf(slt)
class(slt)
```

```
## [1] "sf"         "grouped_df" "tbl_df"     "tbl"        "data.frame"
```

```r
slt <- st_simplify(slt)
```

- Add the Williamette and Columbia Rivers shape files.

```r
rivers <- st_read("data/RIVERS/mjriv_fi/mjriv_fi.shp")
```

```
## Reading layer `mjriv_fi' from data source `C:\Users\aloosefish\Documents\GeoData\GIS1_LabData\Term_pa
## Simple feature collection with 23 features and 2 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 7499924 ymin: 528102.1 xmax: 7876784 ymax: 877995.8
## epsg (SRID):    NA
## proj4string:    +proj=lcc +lat_1=44.33333333333334 +lat_2=46 +lat_0=43.66666666666666 +lon_0=-120.5 +
```

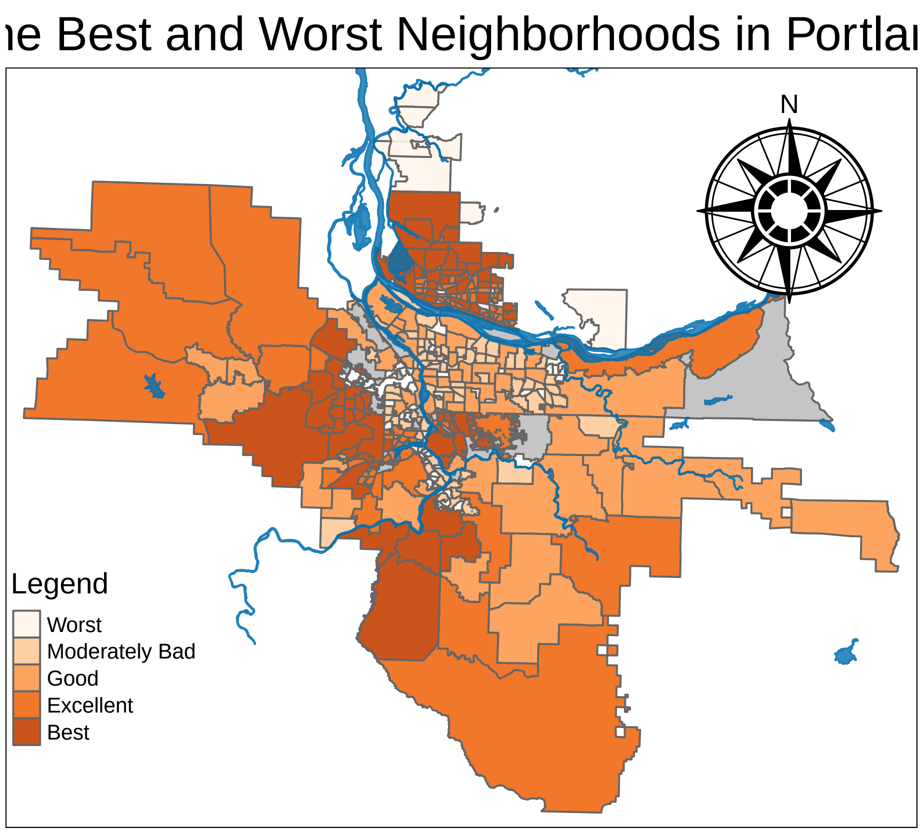
```r
the_crs <- st_crs(slt)
st_transform(rivers, the_crs)
```

```
## Simple feature collection with 23 features and 2 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 7499924 ymin: 528102.1 xmax: 7876784 ymax: 877995.8
## epsg (SRID):    NA
## proj4string:    +proj=lcc +lat_1=44.33333333333334 +lat_2=46 +lat_0=43.66666666666666 +lon_0=-120.5 +
## First 10 features:
##    CLASS         AREA                       geometry
## 1      1 2.658044e+05 POLYGON ((7634596 640411.9,...
## 2      1 1.286660e+04 POLYGON ((7633834 640440.1,...
## 3      1 7.969020e+03 POLYGON ((7645819 643750, 7...
## 4      1 1.436035e+02 POLYGON ((7645059 645240.4,...
## 5      1 1.326917e+06 POLYGON ((7645059 645240.4,...
## 6      1 1.875614e+07 POLYGON ((7858933 657351, 7...
## 7      1 1.510992e+07 POLYGON ((7786130 655798.8,...
## 8      1 1.663599e+07 POLYGON ((7808595 669317.6,...
## 9      1 5.450561e+05 POLYGON ((7609603 765441.1,...
## 10     1 1.985069e+09 POLYGON ((7794584 703631.3,...
```

```r
#rivers <- st_simplify(rivers)
# test map
# tm_shape(rivers, unit = "mi") +
#  tm_polygons(col = "#08306B")
```

The Map!

```
tm_shape(slt, unit = "mi") +
    tm_layout(main.title = "The Best and Worst Neighborhoods in Portland", main.title.position = "cente
    tm_polygons(col = "index_score", palette = "Oranges", style = "quantile", n = 5, labels = c("Worst"
     tm_shape(rivers, unit = "mi") +
  tm_polygons(col = "#0571b0", alpha = 0.8, border.alpha = 0.9, border.col = "#0571b0") +
    tm_compass(type = "rose", position = c("right", "top")) +
    tm_legend() #+
```




```
    #tm_credits("Ian Davis\nJune 12, 2018\nGIS 1, Term Project\nData sources: Oregon Metro RLIS,\nPSU D
```

Best Neighborhoods

```
top_10
```

```
## # A tibble: 352 x 2
## # Groups:   NAME [352]
##    NAME                               index_score
##    <fct>                                    <dbl>
##  1 SOUTH CANBY                              6484.
##  2 CANBY                                    2304.
##  3 AURORA - BUTTEVILLE - BARLOW             2220.
##  4 CARUS                                    1725.
##  5 CPO 4M METZGER                           1336.
```

```
##   6 WEST HAZEL DELL                        1301.
##   7 CPO 4B BULL MTN                        1269.
##   8 CENTRAL POINT - LELAND - NEW ERA       1214.
##   9 FAIRGROUNDS                            1214.
##  10 FRUIT VALLEY                           1176.
## # ... with 342 more rows
```

Worst Neighborhoods

```
bottom_10
```

```
## # A tibble: 352 x 2
## # Groups:   NAME [352]
##    NAME                       index_score
##    <fct>                            <dbl>
##  1 RIDGEFIELD JUNCTION           -0.0331
##  2 WASHOUGAL RIVER               -0.0228
##  3 MEADOW GLADE                  -0.00798
##  4 ENTERPRISE/PARADISE POINT     -0.00456
##  5 E. FORK FRONTIER              -0.00342
##  6 NORTH FORK LEWIS RIVER        -0.00342
##  7 EAST FORK HILLS               -0.00342
##  8 BOISE/ELIOT                    0.0940
##  9 ALAMEDA-BEAUMONT-WILSHIRE      0.148
## 10 PORTLAND UNCLAIMED #5          0.200
## # ... with 342 more rows
```

Data Sources

- Oregon Metro RLIS (shape files for Portland neighborhoods and rivers) http://rlisdiscovery.oregonmetro.gov/?resourceID=1
- Portland State University Department of Geography (real estate data)

Packages

- Tennekes M (2018). "tmap: Thematic Maps in R." *Journal of Statistical Software*, *84*(6), pp. 1-39. doi: 10.18637/jss.v084.i06 (URL: http://doi.org/10.18637/jss.v084.i06).
- Hadley Wickham, Romain Fran?ois, Lionel Henry and Kirill M?ller (2018). dplyr: A Grammar of Data Manipulation. R package version 0.7.6. https://CRAN.R-project.org/package=dplyr
- Edzer Pebesma (2018). sf: Simple Features for R. R package version 0.6-3. https://CRAN.R-project.org/package=sf