Math 161 Spatial Statistics - Final Project

Nate Stringham

4/16/2019

The Goal of this project is to utilize spatial kriging to create heat maps for NBA players.

Research Questions - How does a NBA player's scoring efficiency change with shot location? - Can predict where on the court they are most efficient?

We'll need the following packages to aid us in our analysis. 1. tidyverse - access to many important data wrangling, cleaning, and viz tools. 2. SpatialBall - This package contains shot data for the 2016-17 NBA season including the location of every shot. 3. sp and gstat - Provide methods for implementing spatial kriging.

```
# Dependencies
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(SpatialBall)
library(sp)
library(gstat)
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

First we need to wrangle our data into a format that makes sense for our analysis. We are interested in analyzing the scoring efficiency of various NBA players which is traditionally done with a half court shot chart. We'll need to filter the data to only include shots from halfcourt (we aren't concerned with shots taken from beyond halfcourt) and also select for our player of interest. We'll also add a new feature, points per shot (PPS), to our data frame in order to analyze scoring efficiency.

```
# Player of Interest
player <- "Damian Lillard"

#Filter player's shot data to half court
PlayerShots <- season2017 %>%
  select(PLAYER_NAME,LOC_X, LOC_Y, SHOT_TYPE, SHOT_MADE_FLAG) %>%
  filter(PLAYER_NAME == player, LOC_Y < 500)

#Add columns to show shot type,
# whether it was made/missed, and points per shot
PlayerShots$TYPE <- ifelse(PlayerShots$SHOT_TYPE == "2PT Field Goal", 2, 3)
PlayerShots$Made_Miss <- ifelse(PlayerShots$SHOT_MADE_FLAG == 0, 0, 1)
PlayerShots$PPS <- (PlayerShots$TYPE)*(PlayerShots$Made_Miss)
```

```
head(PlayerShots)
```

```
##      PLAYER_NAME LOC_X LOC_Y      SHOT_TYPE SHOT_MADE_FLAG TYPE Made_Miss
## 1 Damian Lillard   -81   242 3PT Field Goal              1    3         1
## 2 Damian Lillard   104     7 2PT Field Goal              0    2         0
## 3 Damian Lillard    91   144 2PT Field Goal              1    2         1
## 4 Damian Lillard   -22     7 2PT Field Goal              1    2         1
## 5 Damian Lillard  -174    77 2PT Field Goal              0    2         0
## 6 Damian Lillard   -38   249 3PT Field Goal              1    3         1
##   PPS
## 1   3
## 2   0
## 3   2
## 4   2
## 5   0
## 6   3
```

Note that some players have taken multiple shots from the same court location, so we need our PPS value to take this into account for these shot locations.

```
# Find all of the duplicate shot locations
nvals <- PlayerShots %>%
  group_by(LOC_X, LOC_Y) %>%
  count()

# Number of shots taken at each location
head(nvals)
```

```
## # A tibble: 6 x 3
## # Groups:   LOC_X, LOC_Y [6]
##   LOC_X LOC_Y     n
##   <dbl> <dbl> <int>
## 1  -245     2     1
## 2  -243   180     1
## 3  -241     7     1
## 4  -240   110     1
## 5  -240   164     1
## 6  -238    51     1
```

```
#Join nvals to the original table and create pps avg - thin data set to include only distinct values.
PlayerShots2 <- PlayerShots %>%
  left_join(nvals) %>%
  group_by(LOC_X, LOC_Y) %>%
  mutate(ppsum = sum(PPS), ppsavg = ppsum/n) %>%
  distinct(LOC_X, LOC_Y, .keep_all = TRUE)
```

```
## Joining, by = c("LOC_X", "LOC_Y")
```

```
head(PlayerShots2)
```

```
## # A tibble: 6 x 11
## # Groups:   LOC_X, LOC_Y [6]
##   PLAYER_NAME LOC_X LOC_Y SHOT_TYPE SHOT_MADE_FLAG  TYPE Made_Miss   PPS
##   <fct>       <dbl> <dbl> <fct>     <fct>          <dbl>     <dbl> <dbl>
## 1 Damian Lil~   -81   242 3PT Fiel~ 1                  3         1     3
## 2 Damian Lil~   104     7 2PT Fiel~ 0                  2         0     0
```

```
## 3 Damian Lil~    91   144 2PT Fiel~ 1                   2         1     2
## 4 Damian Lil~   -22     7 2PT Fiel~ 1                   2         1     2
## 5 Damian Lil~  -174    77 2PT Fiel~ 0                   2         0     0
## 6 Damian Lil~   -38   249 3PT Fiel~ 1                   3         1     3
## # ... with 3 more variables: n <int>, ppsum <dbl>, ppsavg <dbl>
```

Now that we have our feature of interest (PPS) it's time to create some spatial objects so that we can perform the spatial analysis. We'll need the following objects: - spatial points dataframe - prediction grid

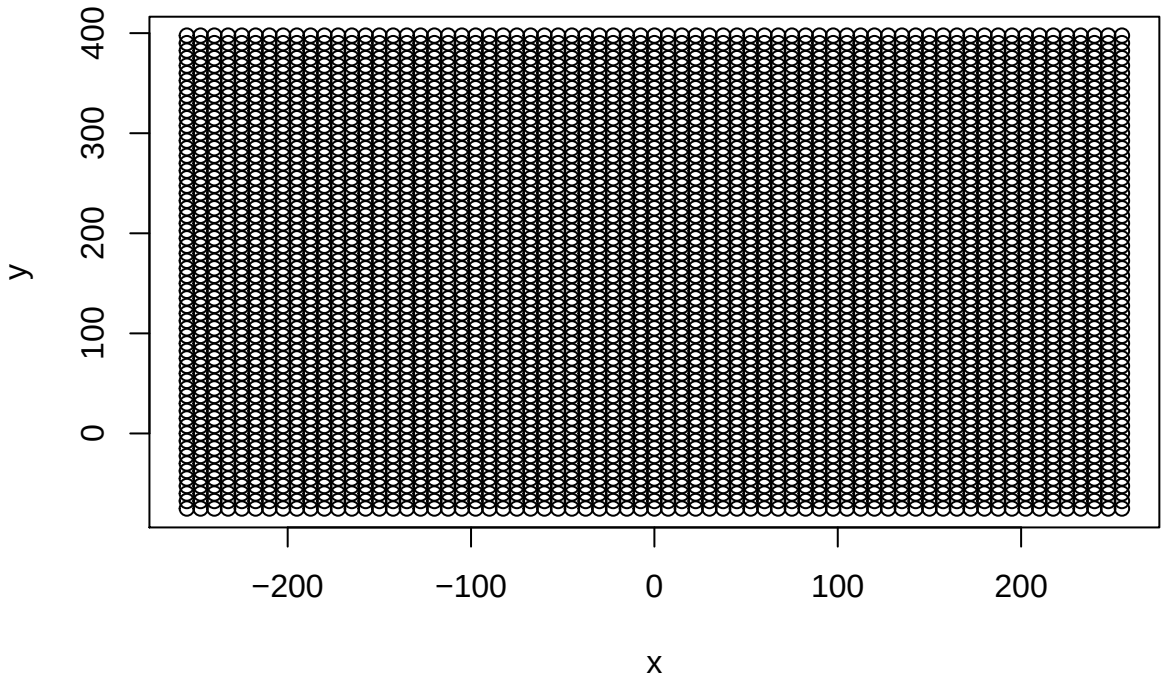
```
# Create a spatial points dataframe from shot locations
coordinates(PlayerShots2)<- ~LOC_X + LOC_Y
class(PlayerShots2)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
#Make the spatial grid for predictions
grid <- expand.grid(x = seq(-255, 255, by = 7.5), y = seq(-75, 400, by= 7.5))
class(grid)
```

```
## [1] "data.frame"
```

```
plot(grid)
```




```
coordinates(grid)<- ~x+y
class(grid)
```

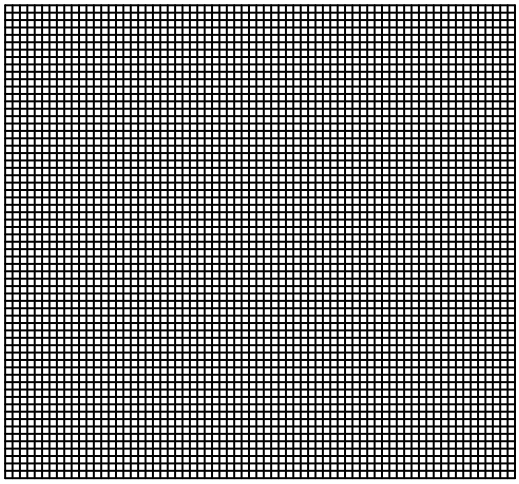
```
## [1] "SpatialPoints"
## attr(,"package")
## [1] "sp"
```

```
courtgrid <- SpatialPixels(grid)
class(courtgrid)
```

```
## [1] "SpatialPixels"
```

```
## attr(,"package")
## [1] "sp"
```

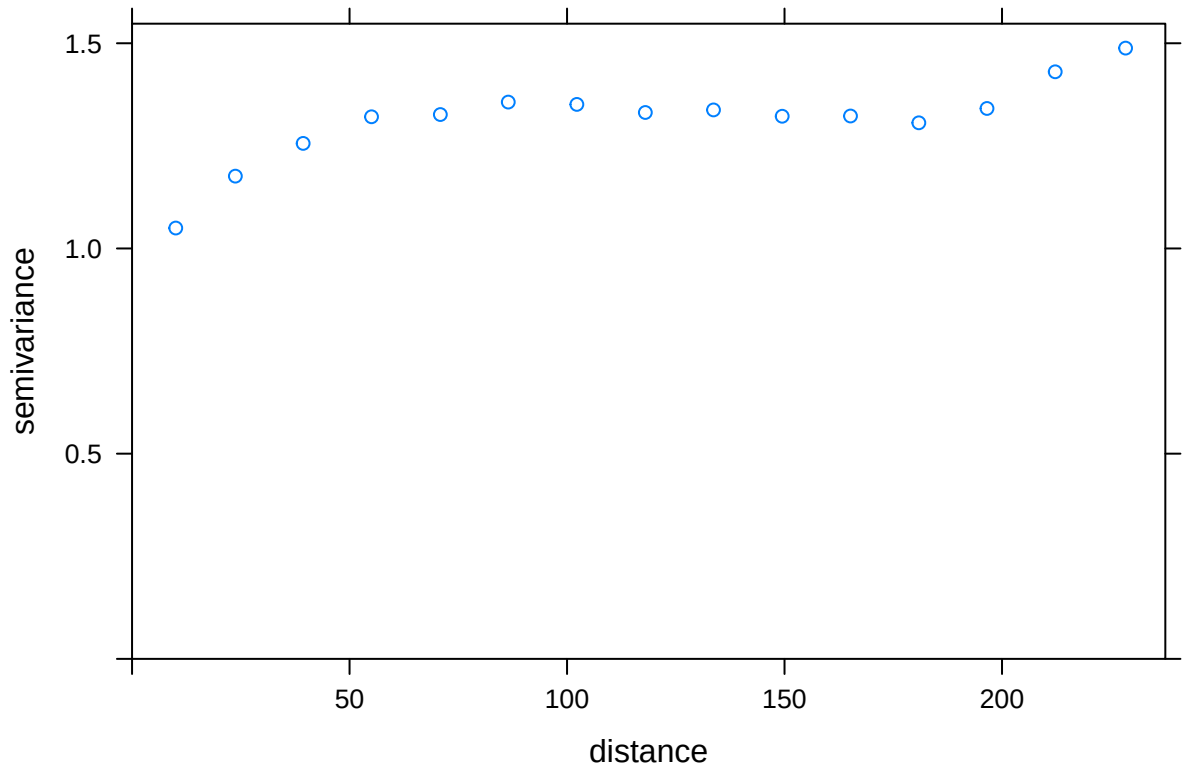
```
plot(courtgrid)
```



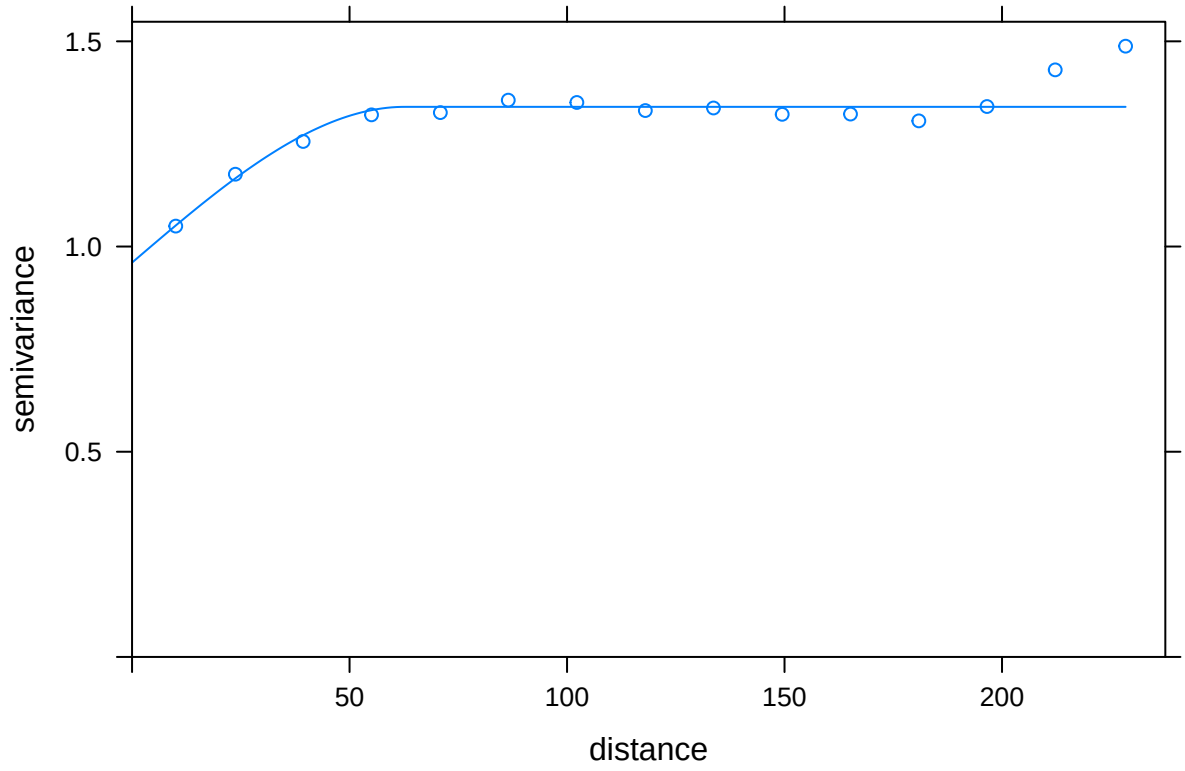
One of the advantages of Spatial Kriging is it takes into account the variation of our feature of interest as a function of distance. Thus, we first need to capture this variation by building a sample variogram (variogram cloud is also helpful to gain understanding about this variation)

```
# Plot Variogram cloud
#vargram.cloud <- gstat::variogram(ppsavg~1, data = PlayerShots2, cloud = TRUE)
#plot(vargram.cloud)

# Plot variogram
vargram <- gstat::variogram(ppsavg~1, data = PlayerShots2)
plot(vargram)
```




```
# find the best fit for the variogram
fit <- fit.variogram(vargram, vgm("Sph", "Mat", "Exp"))
plot(vargram, model = fit)
```

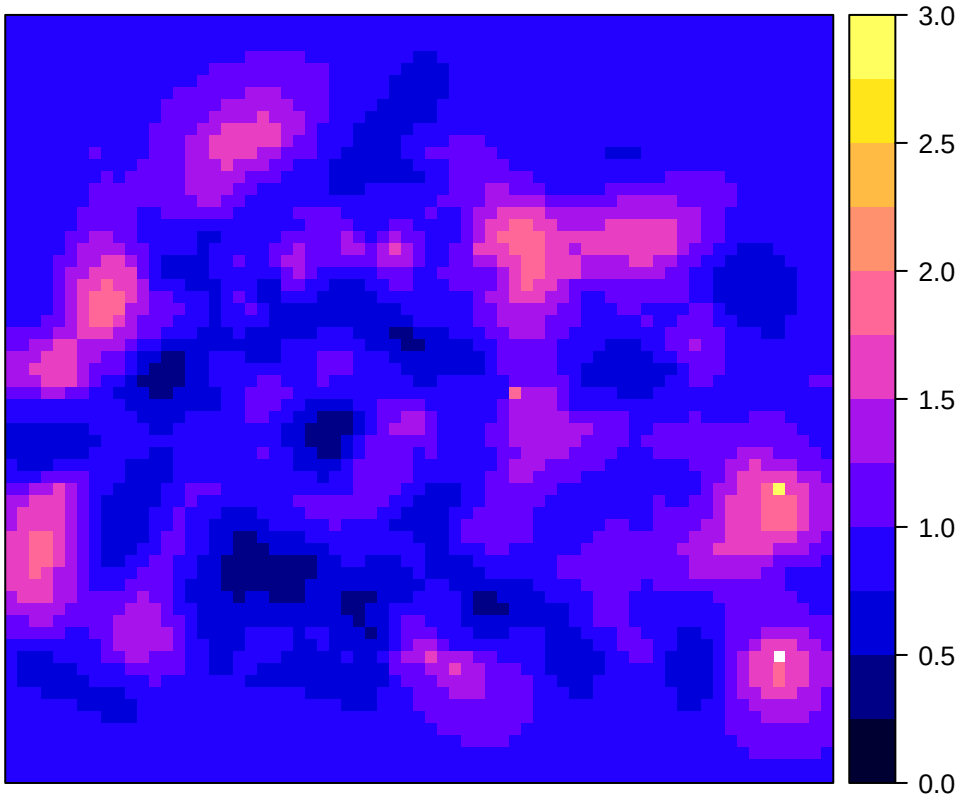


Now we are ready to perform the Kriging and visualize the results!

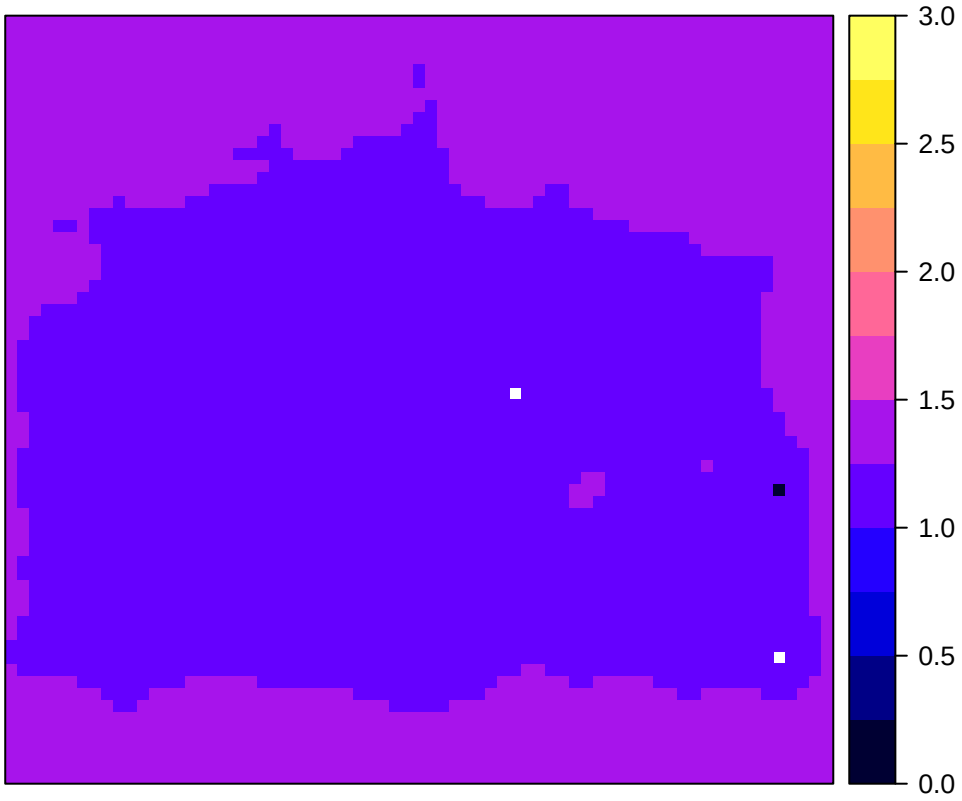
```r
# Make Kriged predictions
z.krige <- gstat::krige(ppsavg~1, PlayerShots2, courtgrid, model = fit)
```

```
## [using ordinary kriging]
```

```r
# Visualize
spplot(z.krige["var1.pred"], at = seq(0,3, by =.25))
```




```r
spplot(z.krige["var1.var"], at = seq(0,3, by = .25))
```



As a final step, we can run a Moran's I test to get a feel for the amount of spatial autocorrelation present.

```
library(spdep)
```

```
## Loading required package: spData
```

```
## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`
```

```
## Loading required package: sf
```

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

```
# Run Moran's I

#Create neighbor list (graph-based since working with points)
grph <- relativeneigh(PlayerShots2)
neib <- graph2nb(grph)
neib.listw <- nb2listw(neib, style="B", zero.policy = TRUE)


mtest <- moran.test(PlayerShots2@data$ppsavg, listw=neib.listw, alternative="two.sided", zero.policy = 
mtest
```

```
##
##  Moran I test under randomisation
##
## data:  PlayerShots2@data$ppsavg
## weights: neib.listw  n reduced by no-neighbour observations
##
##
```

```
## Moran I statistic standard deviate = 1.1125, p-value = 0.2659
## alternative hypothesis: two.sided
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.0261992965     -0.0010989011      0.0006020485
```

```
sim1 <- moran.mc(PlayerShots2@data$ppsavg, listw=neib.listw, nsim=99, zero.policy = TRUE, alternative="l
sim1
```

```
##
##  Monte-Carlo simulation of Moran I
##
## data:  PlayerShots2@data$ppsavg
## weights: neib.listw
## number of simulations + 1: 100
##
## statistic = 0.026199, observed rank = 97, p-value = 0.97
## alternative hypothesis: less
```

After running Moran's I we see that the p-value fairly high and the Moran's I statistic very close to zero meaning that there is not strong positive spatial autocorrelation.