geospatial_project.R

Jaspo

Wed Mar 20 19:51:22 2019

```
library(geoR)
```

```
## ----------------------------------------------------------
##  Analysis of Geostatistical Data
##  For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
##  geoR version 1.7-5.2.1 (built on 2016-05-02) is now loaded
## ----------------------------------------------------------
```

```
library(sp)
library(gstat)
library(maps)
library(ggplot2)
library(lattice)

### This geospatial project is based on a AirBnb dataset that looks at average bookin
g costs per night
### on AirBnb in Texas



air <- read.csv("C:/Users/Jaspo/Documents/geostats/Air.csv", header = T)

### First thing we want to check are amount of NA values within each column

colSums(is.na(air))
```

```
##                       X average_rate_per_night         bedrooms_count
##                       0                       0                      0
##                    city        date_of_listing            description
##                       0                       0                      0
##               latitude              longitude                  title
##                      34                      34                      0
##                     url
##                       0
```

```
### Looking at structure of data to see which variables are categorical, and which are numeric

str(air)
```

```
## 'data.frame':    18259 obs. of  10 variables:
##  $ X                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ average_rate_per_night: Factor w/ 701 levels "","$10","$100",..: 268 101 541 54 8 613 245 63 244 351 600 ...
##  $ bedrooms_count       : Factor w/ 14 levels "","1","10","11",..: 6 8 2 2 6 8 7 2 7 14 ...
##  $ city                 : Factor w/ 505 levels "Abilene","Addison",..: 244 411 24 1 68 186 121 88 186 399 411 ...
##  $ date_of_listing      : Factor w/ 102 levels "April 2009","April 2010",..: 77 80 43 34 35 17 68 42 34 14 ...
##  $ description          : Factor w/ 11077 levels "","'Perfect Escape' Duplex Historic Grapevine\\n\\nThis charming, newly renovated, fully-furnished two bedroom, tw"| __truncated__,..: 10676 8254 3 7292 10650 5301 7859 9354 3043 7344 ...
##  $ latitude             : num  30 29.5 29.8 30.6 32.7 ...
##  $ longitude            : num  -95.3 -98.4 -95.1 -96.3 -97.3 ...
##  $ title                : Factor w/ 11399 levels "","'66 MCM MadMen Private Apt /courtyard /pool",..: 426 10821 8711 8035 10431 4892 3420 4649 771 3584 ...
##  $ url                  : Factor w/ 18259 levels "https://www.airbnb.com/rooms/10010566?location=Boerne%2C%20TX",..: 11803 9818 8586 1219 9528 4208 1767 12829 9996 6484 ...
```

```
# Need to do the following conversions:
# average_rate(Factor) -> numeric
# bedrooms_count(Factor) -> numeric


levels(air$bedrooms_count)
```

```
##  [1] ""       "1"      "10"     "11"     "13"     "2"      "3"
##  [8] "4"      "5"      "6"      "7"      "8"      "9"      "Studio"
```

```
table(air$bedrooms_count)
```

```
##
##             1     10     11     13      2      3      4      5      6
##      3   9394     13      1      1   3302   2732   1238    410     76
##      7      8      9 Studio
##     36     22      5   1026
```

```
air$bedrooms_count <- as.character(air$bedrooms_count)
air$bedrooms_count[air$bedrooms_count == "Studio"] <- "0"
air$bedrooms_count[air$bedrooms_count == ""] <- "NA"
air$bedrooms_count <- as.factor(air$bedrooms_count)
table(air$bedrooms_count)
```

```
##
##    0    1   10   11   13    2    3    4    5    6    7    8    9   NA
## 1026 9394   13    1    1 3302 2732 1238  410   76   36   22    5    3
```

```
levels(air$bedrooms_count) <- c("0", "1", "10", "11", "13", "2", "3", "4", "5", "6",
"7", "8", "9", "NA")
air$bedrooms_count <- as.numeric(levels(air$bedrooms_count)[air$bedrooms_count])
```

```
## Warning: NAs introduced by coercion
```

```
# Bedroom_count has successfully been converted to numeric
table(air$bedrooms_count)
```

```
##
##    0    1    2    3    4    5    6    7    8    9   10   11   13
## 1026 9394 3302 2732 1238  410   76   36   22    5   13    1    1
```

```
# Now we focus on changing average_rate from factor to numeriv. The problem here are t
he dollar signs.
# We use gsub in conjuction with as.numeric to get rid of those dollar signs and conve
rt to numeric.

air$average_rate_per_night <- as.numeric(gsub("\\$", "", air$average_rate_per_night))
str(air)
```

```
## 'data.frame':    18259 obs. of  10 variables:
##  $ X                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ average_rate_per_night: num  27 149 59 60 75 250 129 25 345 72 ...
##  $ bedrooms_count       : num  2 4 1 1 2 4 3 1 3 0 ...
##  $ city                 : Factor w/ 505 levels "Abilene","Addison",..: 244 411 24
1 68 186 121 88 186 399 411 ...
##  $ date_of_listing      : Factor w/ 102 levels "April 2009","April 2010",..: 77 8
0 43 34 35 17 68 42 34 14 ...
##  $ description          : Factor w/ 11077 levels "","'Perfect Escape' Duplex Histo
ric Grapevine\\n\\nThis charming, newly renovated, fully-furnished two bedroom, tw"| _
_truncated__,..: 10676 8254 3 7292 10650 5301 7859 9354 3043 7344 ...
##  $ latitude             : num  30 29.5 29.8 30.6 32.7 ...
##  $ longitude            : num  -95.3 -98.4 -95.1 -96.3 -97.3 ...
##  $ title                : Factor w/ 11399 levels "","'66 MCM MadMen Private Apt /c
ourtyard /pool",..: 426 10821 8711 8035 10431 4892 3420 4649 771 3584 ...
##  $ url                  : Factor w/ 18259 levels "https://www.airbnb.com/rooms/100
10566?location=Boerne%2C%20TX",..: 11803 9818 8586 1219 9528 4208 1767 12829 9996 648
4 ...
```

```
colSums(is.na(air))
```

```
##                      X average_rate_per_night         bedrooms_count
##                      0                       28                      3
##                   city        date_of_listing            description
##                      0                        0                      0
##               latitude              longitude                  title
##                     34                       34                      0
##                    url
##                      0
```

```
index <- which(is.na(air$latitude)) # Checking which observations for latitude are NA
air <- air[-index,] # Getting rid of NA rows for latitude
colSums(is.na(air)) # No more NA values for latitude, longtude, and average.. Still 3
NA values for bedrooms_count.
```

```
##                      X average_rate_per_night         bedrooms_count
##                      0                       0                      3
##                   city        date_of_listing            description
##                      0                       0                      0
##               latitude              longitude                  title
##                      0                       0                      0
##                    url
##                      0
```

```
# Should I get rid of these observations as well since we have so many, or try to pred
ict them?
index <- which(is.na(air$bedrooms_count))
air <- air[-index, ]
colSums(is.na(air))
```

```
##                       X average_rate_per_night         bedrooms_count
##                       0                       0                      0
##                    city        date_of_listing            description
##                       0                       0                      0
##                latitude              longitude                  title
##                       0                       0                      0
##                     url
##                       0
```

```
### Data cleansing done!

### Part 2: Spatial Statistics


air_geo <- as.geodata(air[,c(8,7,2)])
```

```
## as.geodata: 6709 replicated data locations found.
##  Consider using jitterDupCoords() for jittering replicated locations.
## WARNING: there are data at coincident or very closed locations, some of the geoR's
functions may not work.
##  Use function dup.coords() to locate duplicated coordinates.
##  Consider using jitterDupCoords() for jittering replicated locations
```

```
### Addressing duplicate coordinates. Will take mean of the duplicate coordinates

d <- data.frame(dup.coords(air_geo))
ind <- as.integer(row.names(d))
d <- data.frame(d, "index" = ind)

length(unique(d$longitude)) # There are 4702 duplicates
```

```
## [1] 4702
```

```
length(unique(d$latitude))
```

```
## [1] 4702
```

```
length(unique(air$longitude))
```

```
## [1] 11513
```

```
length(unique(air$latitude))
```

```
## [1] 11513
```

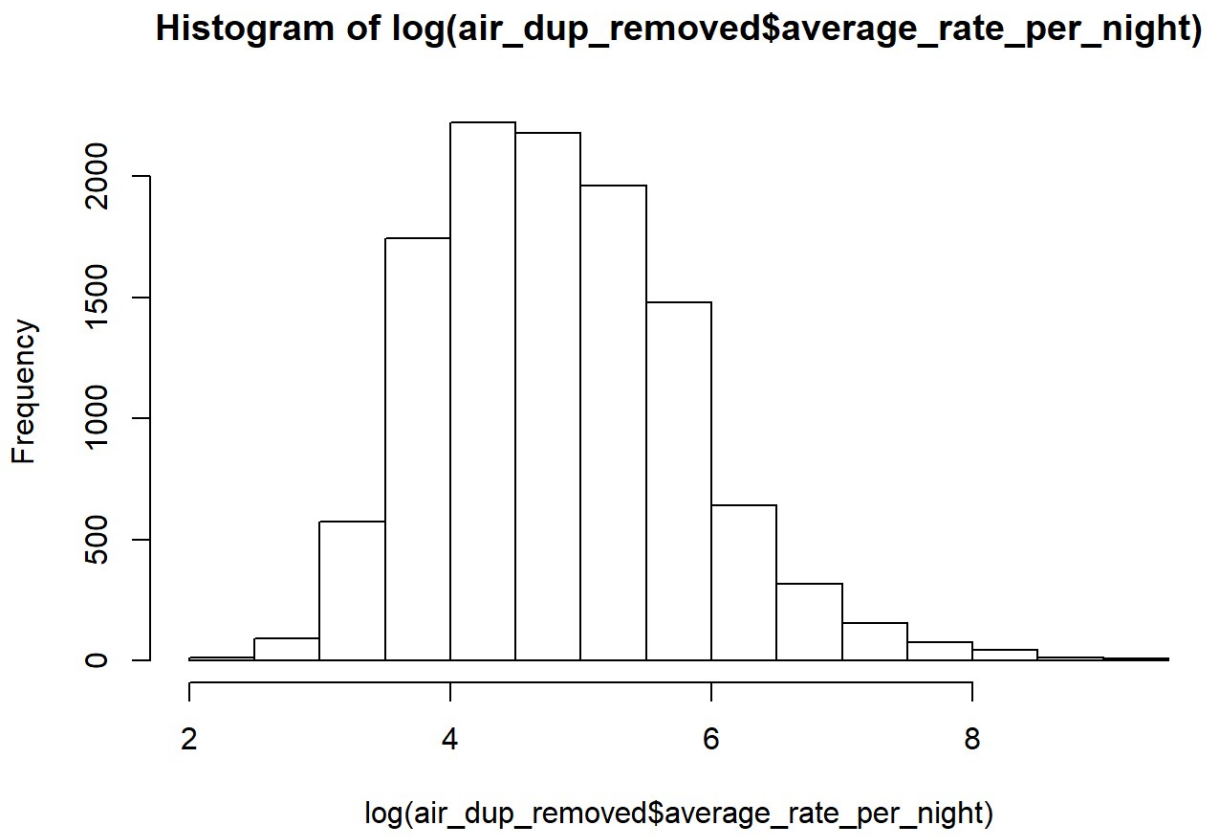
```
test <- data.frame("x" = c(1,2,3,4,5,4,3,2,1))
length(unique(test$x)) # Comes out to 5. So what's happening is it's getting rid of du
plicate,
```

```
## [1] 5
```

```
# but not original
duplicated(test$x)
```

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

```
# We can find mean of duplicate coordinates. Or we can get rid of duplicate points, an
d keep
# original. So limiting AirBnB booking to one intance of one house instead of multipl
e instances
# of one house
dup_ind <- which(duplicated(air$latitude) == TRUE)
air_dup_removed <- air[-dup_ind, ] # duplicates removed in original dataset
hist(log(air_dup_removed$average_rate_per_night))
```



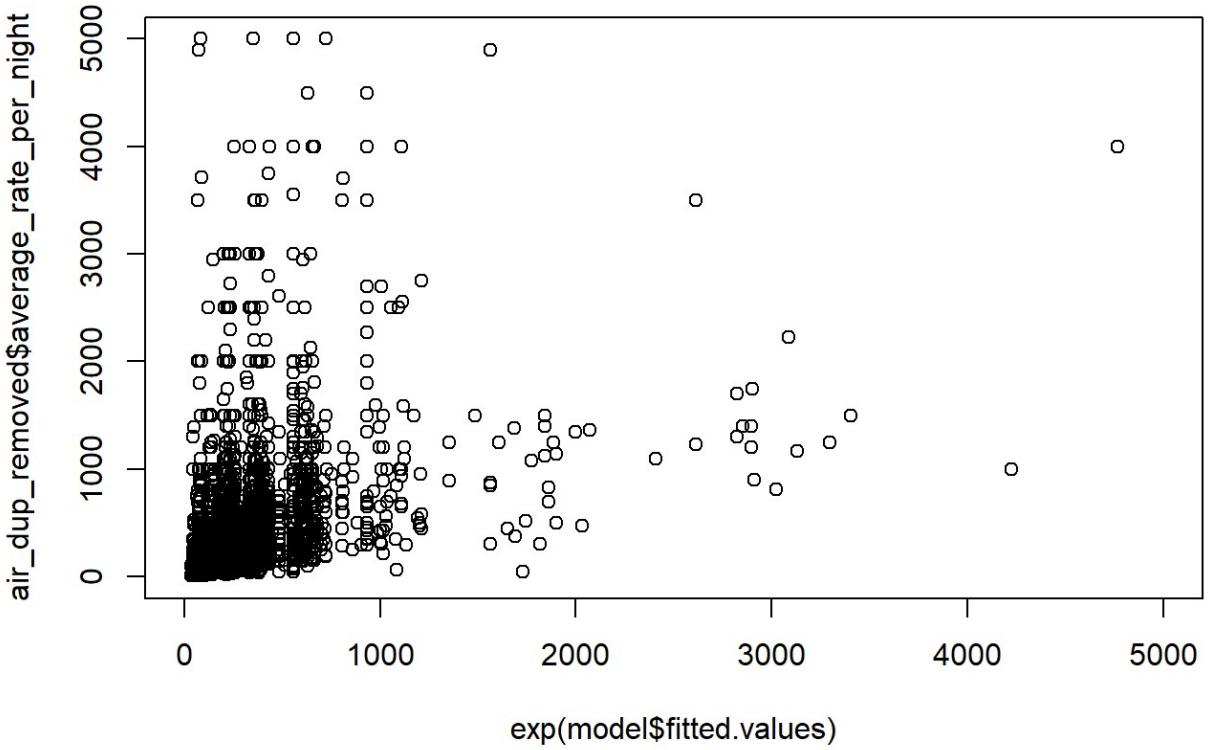

```
### Creating a linear model to see which predictors are significant

model <- lm(log(average_rate_per_night)~bedrooms_count+date_of_listing, data=air_dup_r
emoved)
summary(model)
```

```
##
## Call:
## lm(formula = log(average_rate_per_night) ~ bedrooms_count + date_of_listing,
##     data = air_dup_removed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5444 -0.4969 -0.0208  0.4539  4.5434
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   4.344e+00  7.052e-01   6.160 7.53e-10 ***
## bedrooms_count                5.158e-01  5.328e-03  96.812  < 2e-16 ***
## date_of_listingApril 2010    -8.097e-01  8.637e-01  -0.937    0.349
## date_of_listingApril 2011    -5.351e-01  7.283e-01  -0.735    0.463
## date_of_listingApril 2012    -3.334e-01  7.152e-01  -0.466    0.641
## date_of_listingApril 2013    -4.542e-01  7.120e-01  -0.638    0.523
## date_of_listingApril 2014    -5.232e-01  7.083e-01  -0.739    0.460
## date_of_listingApril 2015    -4.732e-01  7.070e-01  -0.669    0.503
## date_of_listingApril 2016    -5.721e-01  7.063e-01  -0.810    0.418
## date_of_listingApril 2017    -4.050e-01  7.063e-01  -0.573    0.566
## date_of_listingAugust 2009   -2.961e-01  8.143e-01  -0.364    0.716
## date_of_listingAugust 2010   -1.055e+00  7.318e-01  -1.442    0.149
## date_of_listingAugust 2011   -3.726e-01  7.186e-01  -0.519    0.604
## date_of_listingAugust 2012   -4.323e-01  7.103e-01  -0.609    0.543
## date_of_listingAugust 2013   -3.450e-01  7.099e-01  -0.486    0.627
## date_of_listingAugust 2014   -5.648e-01  7.080e-01  -0.798    0.425
## date_of_listingAugust 2015   -5.337e-01  7.067e-01  -0.755    0.450
## date_of_listingAugust 2016   -4.361e-01  7.063e-01  -0.617    0.537
## date_of_listingDecember 2008 -1.017e-12  9.973e-01   0.000    1.000
## date_of_listingDecember 2009  2.612e-01  9.973e-01   0.262    0.793
## date_of_listingDecember 2010  6.900e-02  7.433e-01   0.093    0.926
## date_of_listingDecember 2011 -1.685e-01  7.204e-01  -0.234    0.815
## date_of_listingDecember 2012 -5.700e-01  7.161e-01  -0.796    0.426
## date_of_listingDecember 2013 -4.566e-01  7.104e-01  -0.643    0.520
## date_of_listingDecember 2014 -5.388e-01  7.085e-01  -0.761    0.447
## date_of_listingDecember 2015 -2.297e-01  7.064e-01  -0.325    0.745
## date_of_listingDecember 2016 -3.383e-01  7.066e-01  -0.479    0.632
## date_of_listingFebruary 2009 -5.399e-01  8.143e-01  -0.663    0.507
## date_of_listingFebruary 2010 -7.979e-01  8.143e-01  -0.980    0.327
## date_of_listingFebruary 2011 -4.158e-01  7.269e-01  -0.572    0.567
## date_of_listingFebruary 2012 -3.031e-01  7.130e-01  -0.425    0.671
## date_of_listingFebruary 2013 -3.228e-01  7.101e-01  -0.455    0.649
## date_of_listingFebruary 2014 -3.725e-01  7.083e-01  -0.526    0.599
## date_of_listingFebruary 2015 -4.991e-01  7.079e-01  -0.705    0.481
## date_of_listingFebruary 2016 -4.950e-01  7.063e-01  -0.701    0.483
## date_of_listingFebruary 2017 -4.805e-01  7.068e-01  -0.680    0.497
## date_of_listingJanuary 2010  -9.326e-01  7.885e-01  -1.183    0.237
```

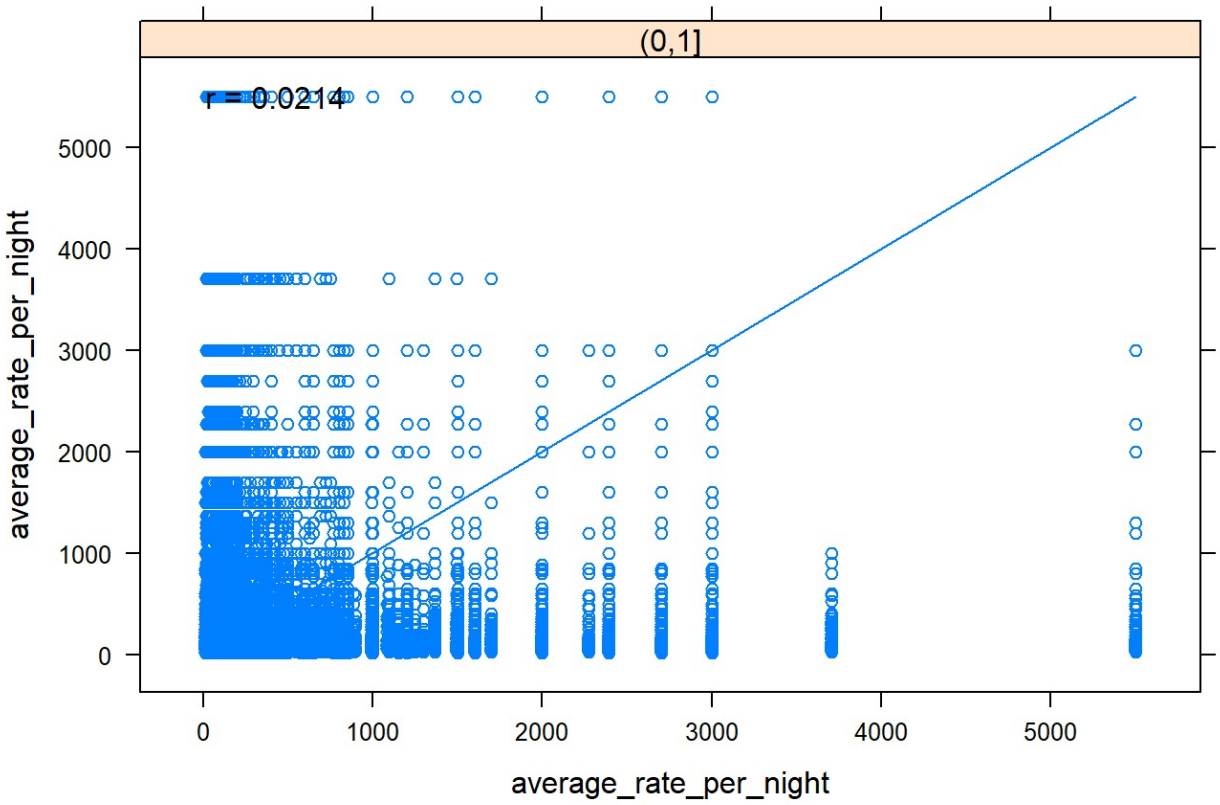
```
## date_of_listingJanuary 2011   -2.606e-02  7.539e-01  -0.035    0.972
## date_of_listingJanuary 2012   -2.965e-01  7.158e-01  -0.414    0.679
## date_of_listingJanuary 2013   -2.222e-01  7.114e-01  -0.312    0.755
## date_of_listingJanuary 2014   -3.434e-01  7.088e-01  -0.485    0.628
## date_of_listingJanuary 2015   -3.703e-01  7.072e-01  -0.524    0.601
## date_of_listingJanuary 2016   -3.555e-01  7.062e-01  -0.503    0.615
## date_of_listingJanuary 2017   -8.616e-02  7.058e-01  -0.122    0.903
## date_of_listingJuly 2009      -6.854e-01  9.973e-01  -0.687    0.492
## date_of_listingJuly 2010      -4.343e-01  7.340e-01  -0.592    0.554
## date_of_listingJuly 2011      -5.283e-01  7.132e-01  -0.741    0.459
## date_of_listingJuly 2012      -3.374e-01  7.177e-01  -0.470    0.638
## date_of_listingJuly 2013      -5.310e-01  7.098e-01  -0.748    0.454
## date_of_listingJuly 2014      -6.524e-01  7.077e-01  -0.922    0.357
## date_of_listingJuly 2015      -4.993e-01  7.063e-01  -0.707    0.480
## date_of_listingJuly 2016      -5.170e-01  7.063e-01  -0.732    0.464
## date_of_listingJune 2009       4.010e-01  9.973e-01   0.402    0.688
## date_of_listingJune 2010      -6.257e-01  7.617e-01  -0.822    0.411
## date_of_listingJune 2011      -4.285e-01  7.218e-01  -0.594    0.553
## date_of_listingJune 2012      -6.980e-01  7.116e-01  -0.981    0.327
## date_of_listingJune 2013      -5.141e-01  7.118e-01  -0.722    0.470
## date_of_listingJune 2014      -5.168e-01  7.081e-01  -0.730    0.465
## date_of_listingJune 2015      -5.285e-01  7.068e-01  -0.748    0.455
## date_of_listingJune 2016      -5.527e-01  7.062e-01  -0.783    0.434
## date_of_listingJune 2017      -4.757e-01  7.066e-01  -0.673    0.501
## date_of_listingMarch 2009     -1.338e+00  9.973e-01  -1.341    0.180
## date_of_listingMarch 2010     -6.854e-01  9.973e-01  -0.687    0.492
## date_of_listingMarch 2011     -3.199e-01  7.158e-01  -0.447    0.655
## date_of_listingMarch 2012     -4.408e-01  7.122e-01  -0.619    0.536
## date_of_listingMarch 2013     -5.760e-01  7.116e-01  -0.810    0.418
## date_of_listingMarch 2014     -4.135e-01  7.081e-01  -0.584    0.559
## date_of_listingMarch 2015     -4.264e-01  7.072e-01  -0.603    0.547
## date_of_listingMarch 2016     -5.298e-01  7.063e-01  -0.750    0.453
## date_of_listingMarch 2017     -6.304e-01  7.066e-01  -0.892    0.372
## date_of_listingMay 2009       -4.913e-01  8.637e-01  -0.569    0.569
## date_of_listingMay 2010       -6.854e-01  9.973e-01  -0.687    0.492
## date_of_listingMay 2011       -1.673e-01  7.138e-01  -0.234    0.815
## date_of_listingMay 2012       -5.259e-01  7.133e-01  -0.737    0.461
## date_of_listingMay 2013       -4.413e-01  7.095e-01  -0.622    0.534
## date_of_listingMay 2014       -4.223e-01  7.078e-01  -0.597    0.551
## date_of_listingMay 2015       -4.410e-01  7.069e-01  -0.624    0.533
## date_of_listingMay 2016       -5.245e-01  7.062e-01  -0.743    0.458
## date_of_listingMay 2017       -4.495e-01  7.063e-01  -0.636    0.525
## date_of_listingNovember 2009   9.996e-02  9.973e-01   0.100    0.920
## date_of_listingNovember 2010  -3.654e-01  7.884e-01  -0.463    0.643
## date_of_listingNovember 2011  -4.236e-01  7.210e-01  -0.587    0.557
## date_of_listingNovember 2012  -4.173e-01  7.204e-01  -0.579    0.562
## date_of_listingNovember 2013  -3.701e-01  7.102e-01  -0.521    0.602
## date_of_listingNovember 2014  -4.771e-01  7.094e-01  -0.673    0.501
## date_of_listingNovember 2015  -4.902e-01  7.066e-01  -0.694    0.488
```

```
## date_of_listingNovember 2016  -4.377e-01  7.067e-01  -0.619    0.536
## date_of_listingOctober 2009   -3.898e-01  8.637e-01  -0.451    0.652
## date_of_listingOctober 2010   -4.194e-01  7.480e-01  -0.561    0.575
## date_of_listingOctober 2011   -6.989e-01  7.218e-01  -0.968    0.333
## date_of_listingOctober 2012   -2.170e-01  7.155e-01  -0.303    0.762
## date_of_listingOctober 2013   -8.478e-01  7.105e-01  -1.193    0.233
## date_of_listingOctober 2014   -3.484e-01  7.073e-01  -0.493    0.622
## date_of_listingOctober 2015   -5.454e-01  7.069e-01  -0.772    0.440
## date_of_listingOctober 2016   -4.985e-01  7.064e-01  -0.706    0.480
## date_of_listingSeptember 2009 -9.722e-01  8.638e-01  -1.126    0.260
## date_of_listingSeptember 2010 -6.375e-01  7.480e-01  -0.852    0.394
## date_of_listingSeptember 2011 -5.508e-01  7.152e-01  -0.770    0.441
## date_of_listingSeptember 2012  2.600e-04  7.114e-01   0.000    1.000
## date_of_listingSeptember 2013 -5.501e-01  7.099e-01  -0.775    0.438
## date_of_listingSeptember 2014 -4.767e-01  7.078e-01  -0.673    0.501
## date_of_listingSeptember 2015 -4.575e-01  7.063e-01  -0.648    0.517
## date_of_listingSeptember 2016 -4.259e-01  7.065e-01  -0.603    0.547
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7052 on 11410 degrees of freedom
## Multiple R-squared:  0.4764, Adjusted R-squared:  0.4717
## F-statistic: 101.8 on 102 and 11410 DF,  p-value: < 2.2e-16
```

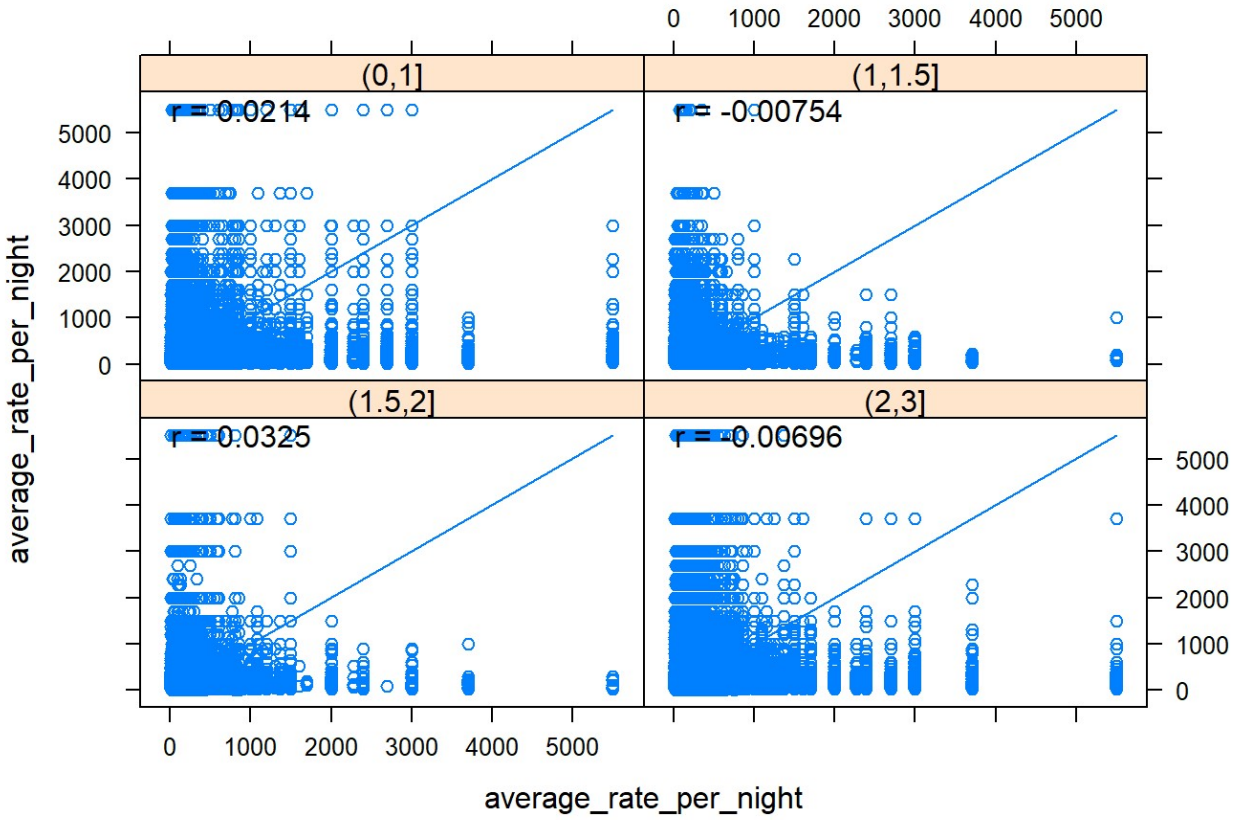
```
### Bedrooms is significant


plot(exp(model$fitted.values), air_dup_removed$average_rate_per_night, xlim=c(0,500
0), ylim=c(0,5000))
```




```
air_dup_removed2 <- air_dup_removed

### Our h-scatterplot

coordinates(air_dup_removed2) <- ~longitude+latitude #Convert the data into spatial da
ta
hscat(average_rate_per_night~1, air_dup_removed2[1:1000,], c(0,1)) #This will produce
the z(s) against z(s+1) plot.
```




```
hscat(average_rate_per_night~1, air_dup_removed2[1:1000,], c(0,1,1.5, 2,3)) #This will
produce 4 different #h-scatterplots with h=1, 2^0.5, 2, 3.
```




```
air_geo_removed <- as.geodata(air_dup_removed[,c(8,7,2)])
dup.coords(air_geo_removed) # successfully removed duplicate coordinates
```

```
## NULL
```

```
new <- data.frame("x" = air_dup_removed[,8], "y" = air_dup_removed[,7], "bedrooms" = a
ir_dup_removed$bedrooms_count, "date" = air_dup_removed$date_of_listing, "data" = air_
dup_removed[,2])
# air_geo_removed1 <- as.geodata(air_dup_removed1)
summary(new)
```

```
##        x                y            bedrooms                 date
##  Min.   :-103.69   Min.   :25.89   Min.   : 0.000   January 2017 : 578
##  1st Qu.: -97.86   1st Qu.:29.63   1st Qu.: 1.000   June 2016    : 365
##  Median : -97.16   Median :30.19   Median : 1.000   January 2016 : 348
##  Mean   : -97.12   Mean   :30.50   Mean   : 1.809   May 2016     : 342
##  3rd Qu.: -96.27   3rd Qu.:32.26   3rd Qu.: 3.000   February 2016: 330
##  Max.   : -93.77   Max.   :35.26   Max.   :13.000   May 2017     : 325
##                                                     (Other)      :9225
##       data
##  Min.   :   10.0
##  1st Qu.:   60.0
##  Median :  118.0
##  Mean   :  226.4
##  3rd Qu.:  229.0
##  Max.   :10000.0
##
```

```
index <- which(new$data >= 1000) # Remmoving outliers (airbnb prices >= 1000)
new1 <- new[-index,]

summary(new)
```

```
##        x                y            bedrooms                 date
##  Min.   :-103.69   Min.   :25.89   Min.   : 0.000   January 2017 : 578
##  1st Qu.: -97.86   1st Qu.:29.63   1st Qu.: 1.000   June 2016    : 365
##  Median : -97.16   Median :30.19   Median : 1.000   January 2016 : 348
##  Mean   : -97.12   Mean   :30.50   Mean   : 1.809   May 2016     : 342
##  3rd Qu.: -96.27   3rd Qu.:32.26   3rd Qu.: 3.000   February 2016: 330
##  Max.   : -93.77   Max.   :35.26   Max.   :13.000   May 2017     : 325
##                                                     (Other)      :9225
##       data
##  Min.   :   10.0
##  1st Qu.:   60.0
##  Median :  118.0
##  Mean   :  226.4
##  3rd Qu.:  229.0
##  Max.   :10000.0
##
```

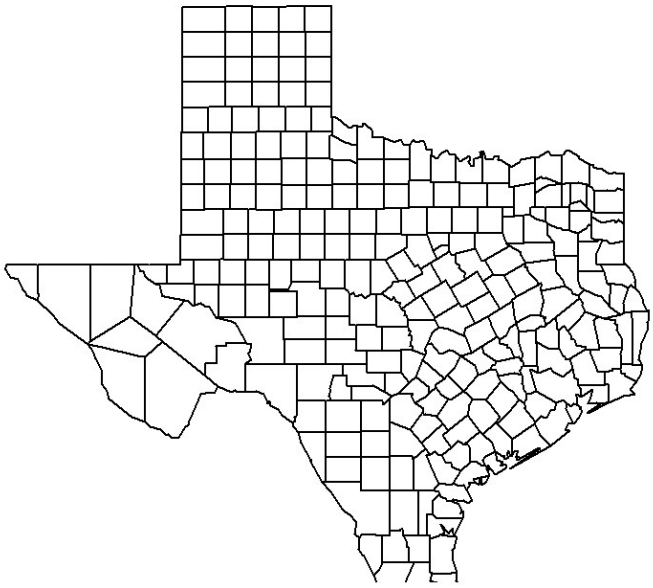
```
set.seed(1)
split <- sample(1:11162, 2500) # My computer kept freezing up with this large of a dat
aset when performing
# kriging. I tried reducing it to 7500, then 5000, then 3500.. Still froze up. Reduce
d to 2500

new2 <- new1[split,]
# Perform universal and ordinary kriging, and compare the two. Do cross vaildation (to
o many points), so
# instead split the data into training and testing. Compare by looking at PRESS of bot
h

# There are 2500 observations. We split the data 70:30
2500*0.70 # 1750 observations will be in training
```

```
## [1] 1750
```

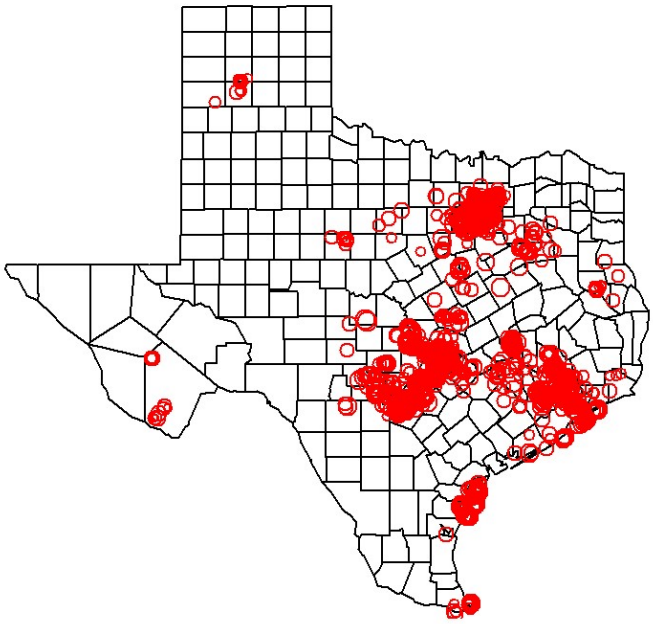
```
2500-1750 # 750 will be in testing (cross-validation)
```

```
## [1] 750
```

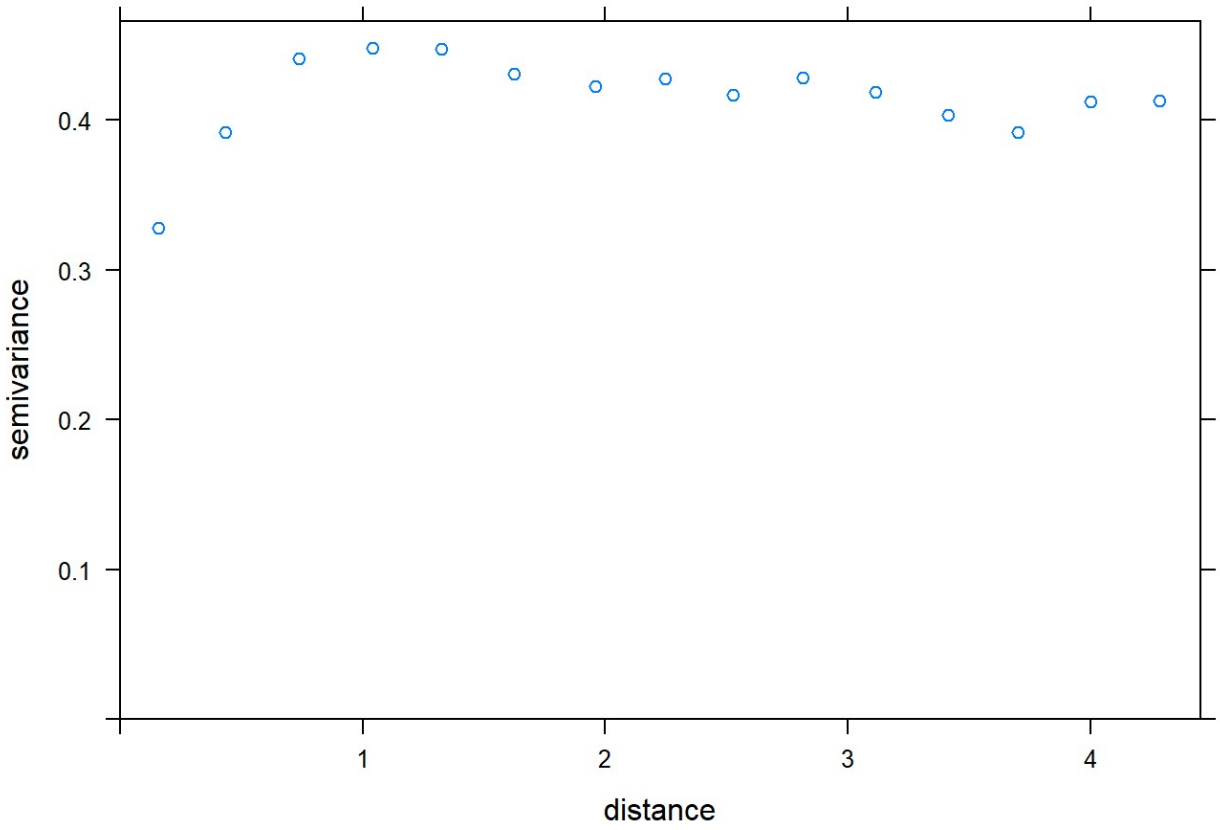
```
set.seed(1)
train <- sample(1:2500, 1750)
model_train <- new2[train,]
model_test <- new2[-train,]

q <- map("county", "texas")
```



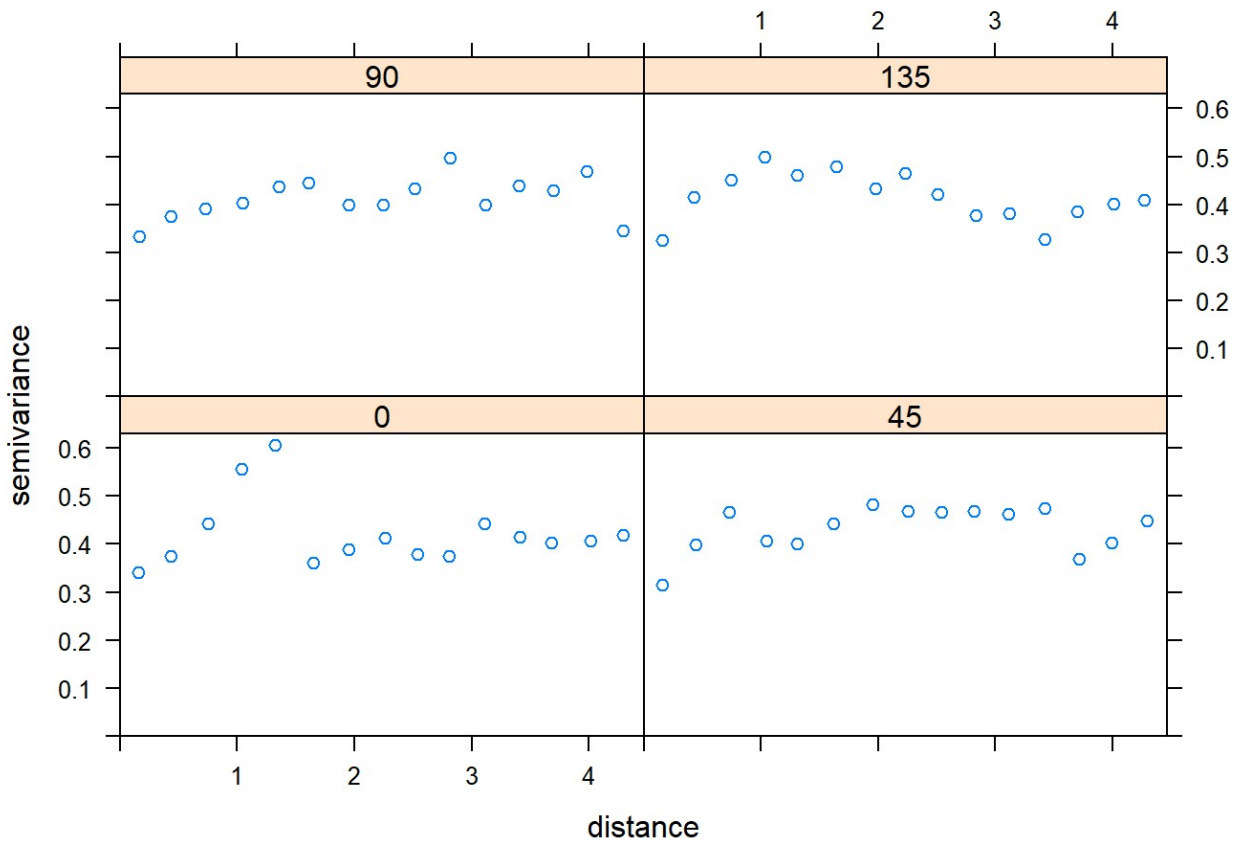
```
map(q)
points(model_train$x, model_train$y, cex=log(model_train$data)/mean(log(model_train$da
ta)), col="red")
```



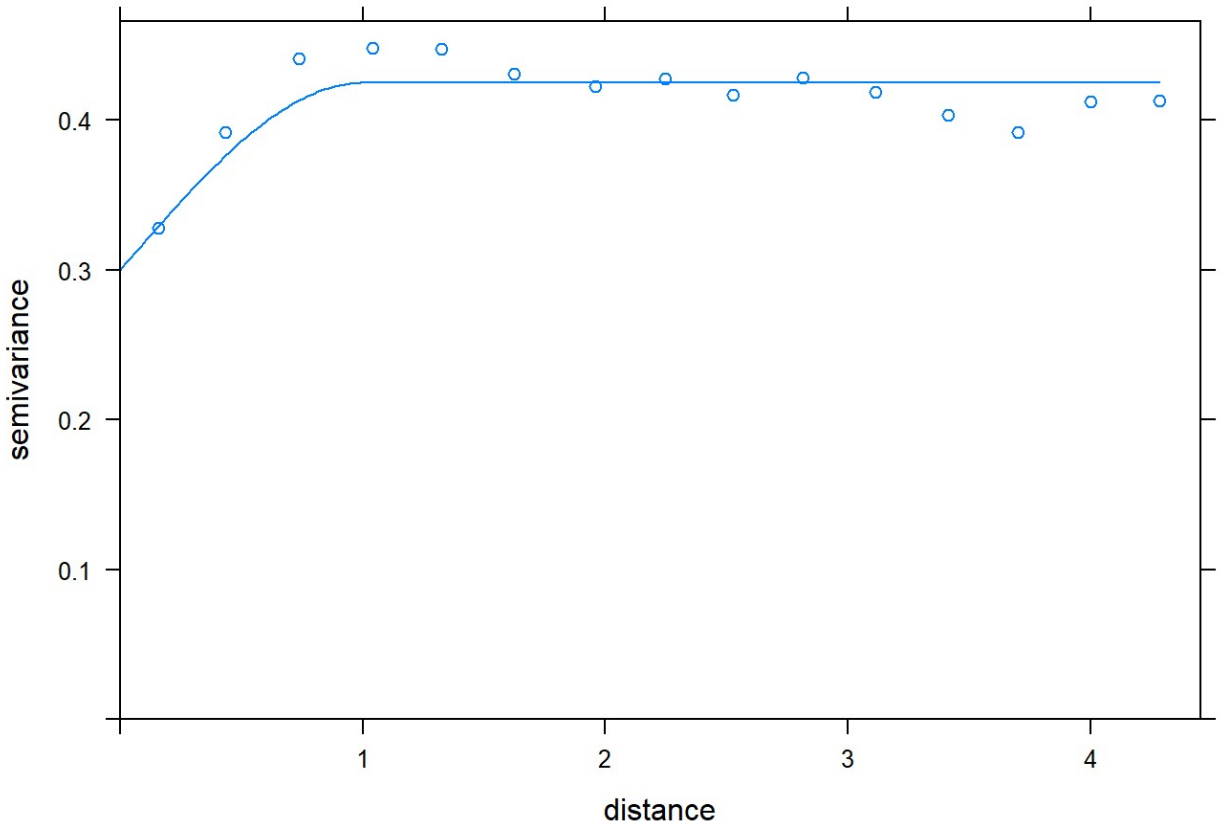
```
### Variogram Modeling

# omnidirectional variogram
g <- gstat(id="log_data", formula = log(data)~model_train$bedrooms, locations = ~x+y,
data = model_train)
q <- variogram(g)
plot(q)
```




```
# variogram in all directions
v <- variogram(g, alpha=c(0,45,90,135))
plot(v)
```



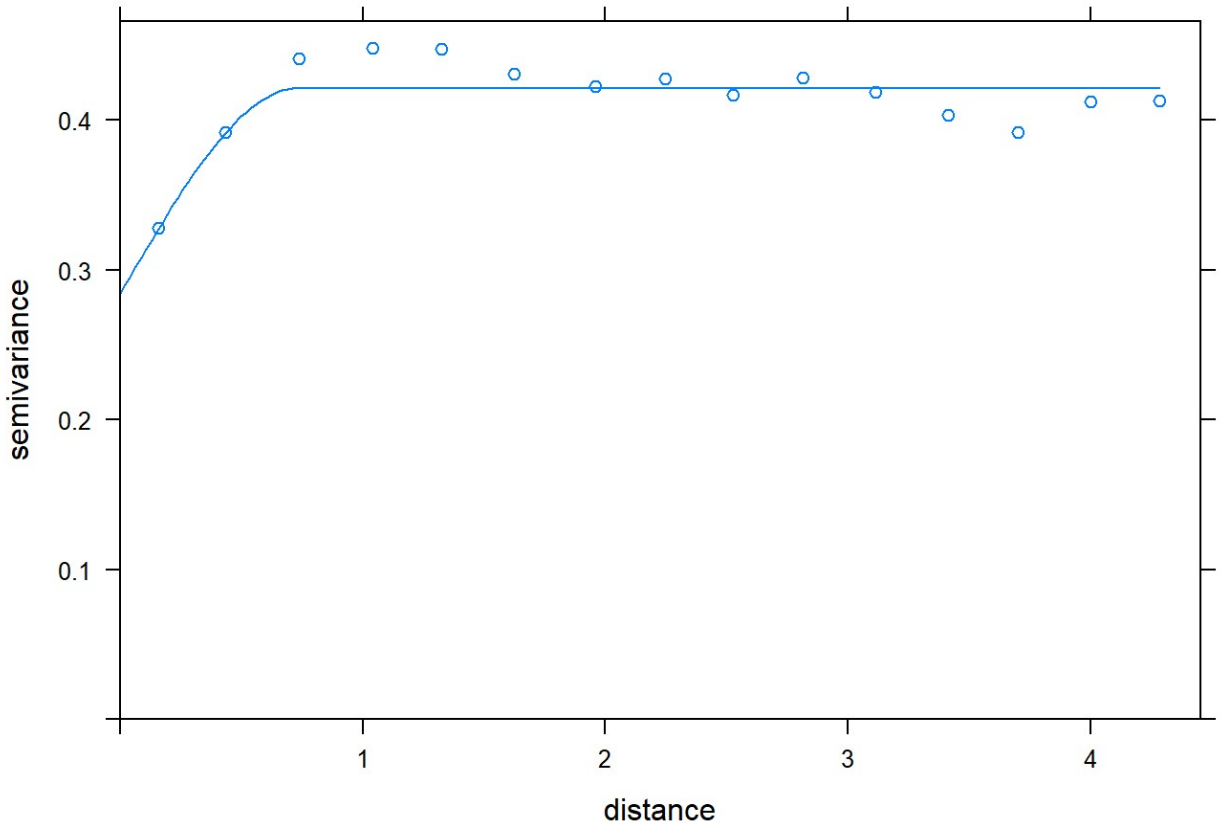

```
# Fitting variogram model
fit_var <- vgm(0.125,"Sph",1.0,0.30)
plot(q, fit_var)
```



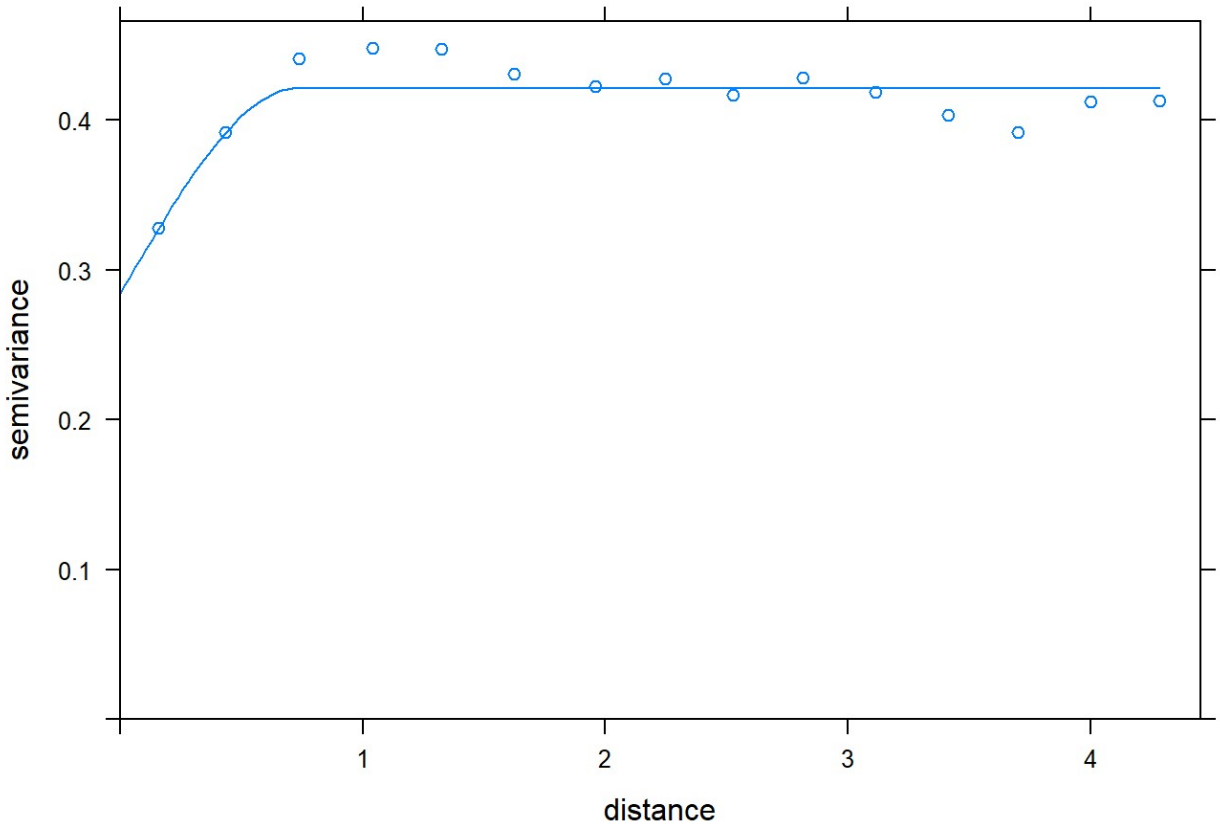

```
# Using Gstat, plot the variogram and fit a variogram model using:
#1) Default weights
#2) Cressie's weights
#3) npairs
#4) OLS

v.fit1 <- fit.variogram(q, vgm(0.125,"Sph",1.0,0.30), fit.method=1) # npairs
v.fit2 <- fit.variogram(q, vgm(0.125,"Sph",1.0,0.30), fit.method=2) # Cressie's
v.fit6 <- fit.variogram(q, vgm(0.125,"Sph",1.0,0.30), fit.method=6) # OLS
```

```
## Warning in fit.variogram(q, vgm(0.125, "Sph", 1, 0.3), fit.method = 6): No
## convergence after 200 iterations: try different initial values?
```

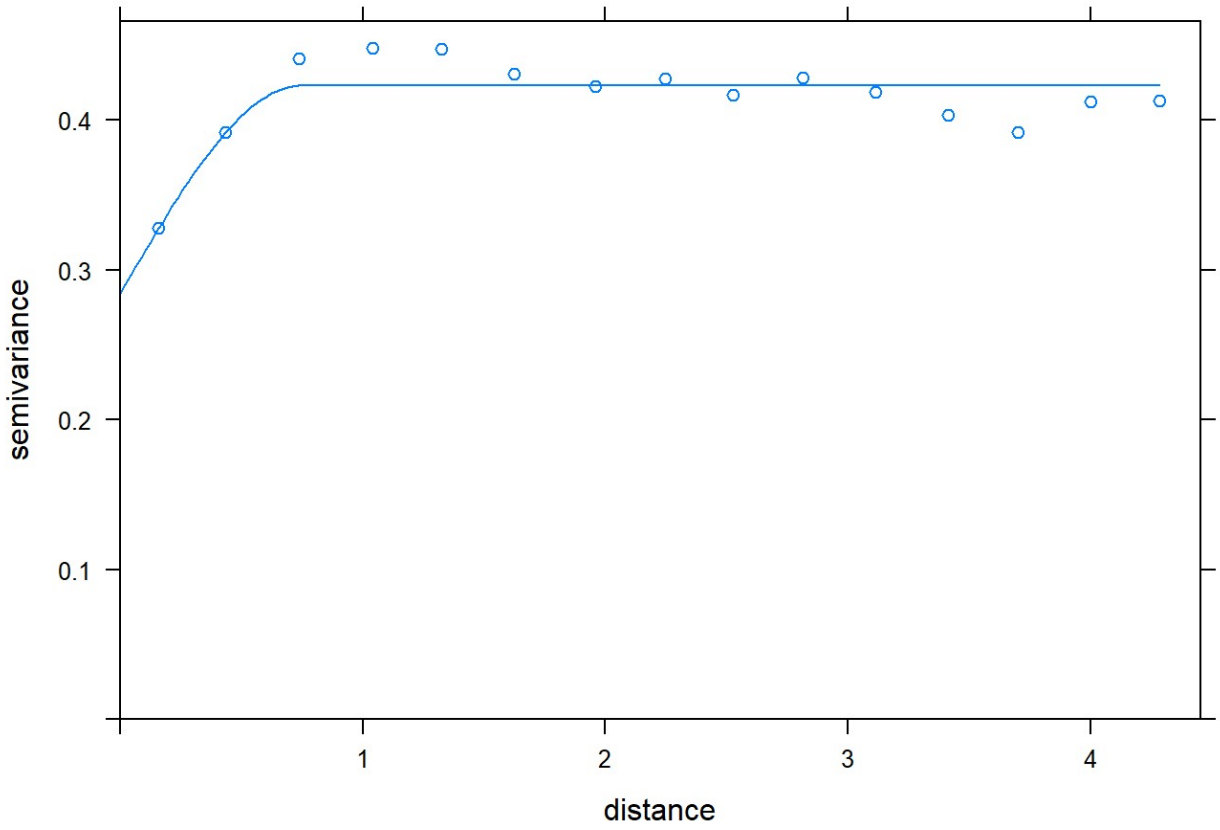
```
v.fit7 <- fit.variogram(q, vgm(0.125,"Sph",1.0,0.30), fit.method=7) # default

# par(mfrow=c(1,1))
plot(q, v.fit1)
```



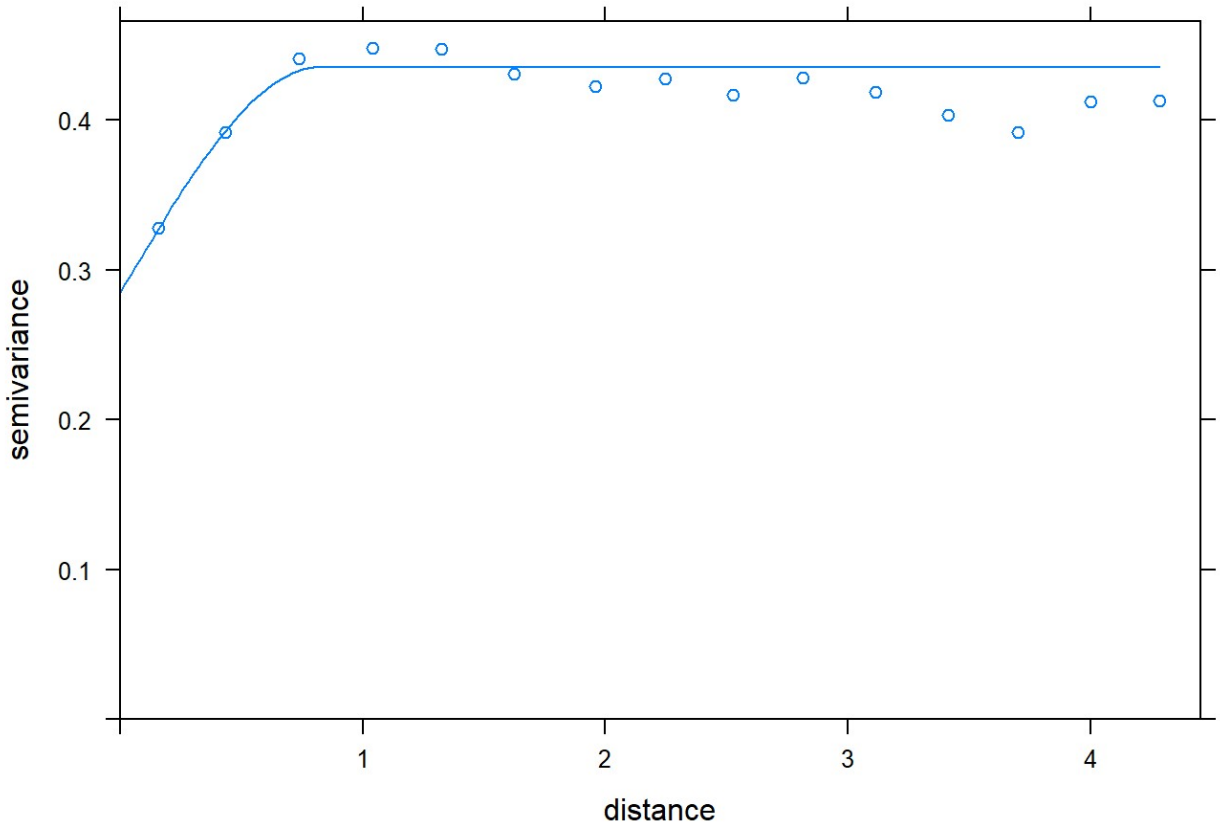

```
plot(q, v.fit2)
```




```
plot(q, v.fit6)
```




```
plot(q, v.fit7)
```




```
# Ordinary Kriging using Cressie's weights
pred <- krige(id="log_data", log(data)~1, locations=~x+y, model=v.fit2, data=model_tra
in, newdata=model_test)
```

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

```
difference <- model_test$data - exp(pred$log_data.pred)
summary(difference)
```

```
##      Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -271.310  -35.580    2.635   50.879   71.613  857.172
```

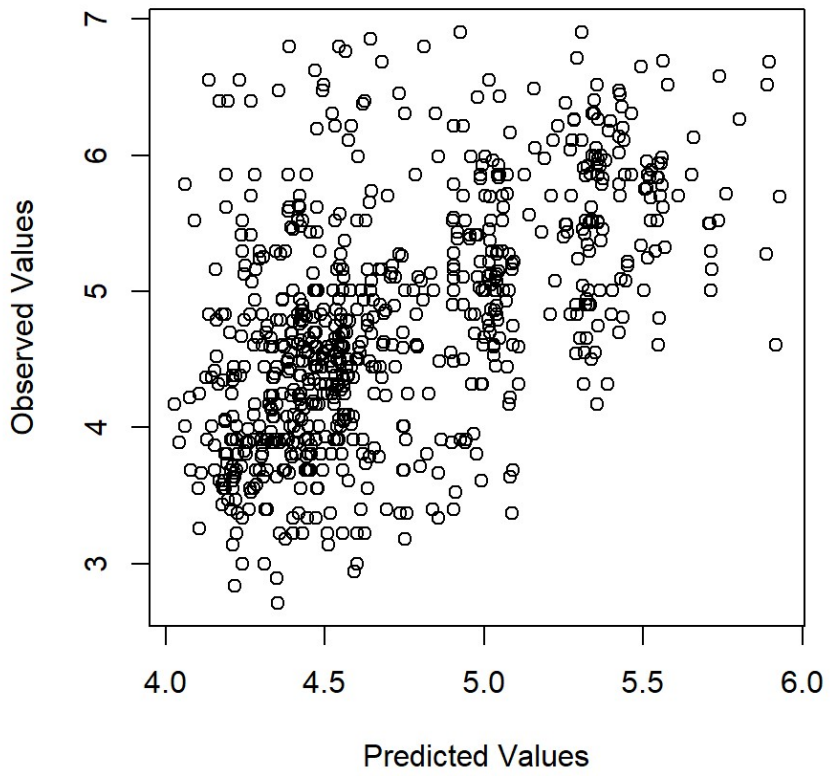
```
cbind(model_test$data[1:100], exp(pred$log_data.pred[1:100]))
```

```
##        [,1]      [,2]
##   [1,]   70 116.83216
##   [2,]  120 155.05129
##   [3,]  197 135.01994
##   [4,]   66  86.70436
##   [5,]  106  94.00988
##   [6,]  250  92.98834
##   [7,]   39  80.56787
##   [8,]  250 250.24985
##   [9,]  250 157.01076
##  [10,]  175 303.34270
##  [11,]  150  87.51358
##  [12,]  475 161.03624
##  [13,]  675 211.71312
##  [14,]   65  87.55533
##  [15,]   99  79.28767
##  [16,]   57  95.44369
##  [17,]   95  81.09241
##  [18,]   40  95.30638
##  [19,]   44  85.00890
##  [20,]   25  78.15836
##  [21,]  234  81.38328
##  [22,]  129  83.86570
##  [23,]  165 120.83970
##  [24,]  165 110.73781
##  [25,]   79  76.70348
##  [26,]  300 148.95270
##  [27,]  350  80.27035
##  [28,]  229  86.01503
##  [29,]   60  95.85671
##  [30,]   68 160.87543
##  [31,]  391 212.52181
##  [32,]  675 360.51246
##  [33,]  160 121.78212
##  [34,]   80  95.33347
##  [35,]   99  95.52674
##  [36,]   68  92.57960
##  [37,]  250  82.66308
##  [38,]  315 245.68498
##  [39,]   38 160.85212
##  [40,]   70  92.04523
##  [41,]  124  84.88657
##  [42,]  450  97.12005
##  [43,]  399 128.39311
##  [44,]   49  78.06820
##  [45,]  150  87.05428
##  [46,]   30 100.83113
##  [47,]  100  78.83985
```

```
##  [48,]   29 116.45046
##  [49,]  350 236.03059
##  [50,]  200  88.64948
##  [51,]   72  93.09759
##  [52,]   44  73.72008
##  [53,]   44 106.64530
##  [54,]  249  83.95354
##  [55,]  225 144.69204
##  [56,]  115 162.66102
##  [57,]  199  69.54328
##  [58,]  105  85.68322
##  [59,]  268  80.29310
##  [60,]  179  94.31484
##  [61,]  185 150.54555
##  [62,]  150  92.49967
##  [63,]  216 150.99112
##  [64,]   77  67.41047
##  [65,]  518 220.36594
##  [66,]  374 205.59812
##  [67,]  140 204.40941
##  [68,]   95  85.56141
##  [69,]  130  89.74524
##  [70,]   89  82.94652
##  [71,]   80  92.93932
##  [72,]  100 119.93051
##  [73,]  450 201.64486
##  [74,]  590 101.33366
##  [75,]  195 359.54762
##  [76,]   40  73.40238
##  [77,]   70 124.86805
##  [78,]   79  62.00827
##  [79,]   70  89.80815
##  [80,]  125  65.20526
##  [81,]   35  71.30313
##  [82,]   35  72.05236
##  [83,]   37  96.84256
##  [84,]  149  88.19902
##  [85,]   55  80.75793
##  [86,]  305 159.52627
##  [87,]  390 152.46931
##  [88,]  380 256.54779
##  [89,]  156 151.87796
##  [90,]  139 104.86708
##  [91,]  303 317.61764
##  [92,]   99 120.52901
##  [93,]  102 108.44081
##  [94,]  126 108.29497
##  [95,]   40 115.12659
##  [96,]   25  68.22047
```

```
##  [97,]  550 235.52546
##  [98,]  367 202.59488
##  [99,]  275 157.51428
## [100,]  163 227.76328
```

```
plot(pred$log_data.pred,log(model_test$data), xlab="Predicted Values", ylab="Observed
Values")
```




```
cor(pred$log_data.pred, log(model_test$data))
```

```
## [1] 0.5416194
```

```
# difference
press <- sum((log(model_test$data) - pred$log_data.pred)^2)
press # PRESS for ordinary kriging is 414.2102
```

```
## [1] 414.2102
```

```
# Universal kriging
pred_uk <- krige(id="log_data", log(data)~bedrooms, locations=~x+y, model=v.fit2, data
=model_train, newdata=model_test)
```

```
## [using universal kriging]
```

```
difference <- model_test$data - exp(pred_uk$log_data.pred)
summary(difference)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1079.823   -24.644     3.109    28.537    52.299   800.904
```

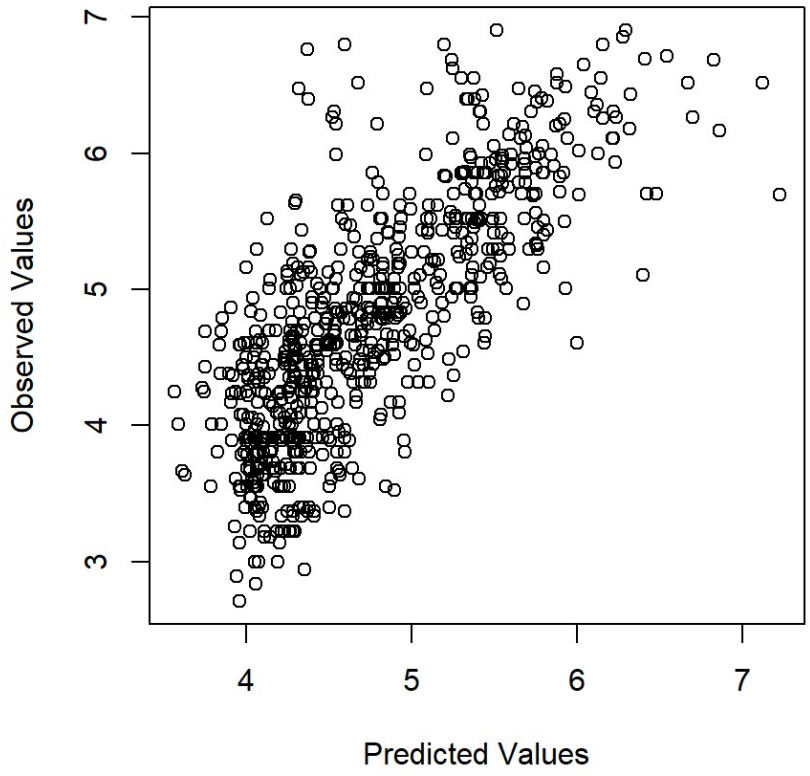
```
cbind(model_test$data[1:100], exp(pred_uk$log_data.pred[1:100]))
```

```
##        [,1]      [,2]
##   [1,]   70  80.96868
##   [2,]  120 123.79614
##   [3,]  197  80.59436
##   [4,]   66  67.62079
##   [5,]  106 109.24394
##   [6,]  250 187.67992
##   [7,]   39  65.34322
##   [8,]  250 162.61461
##   [9,]  250  97.59452
##  [10,]  175 328.84489
##  [11,]  150 122.45252
##  [12,]  475 954.77074
##  [13,]  675 789.42784
##  [14,]   65  77.45400
##  [15,]   99 150.02886
##  [16,]   57 122.71234
##  [17,]   95  83.44021
##  [18,]   40  67.44162
##  [19,]   44  57.34925
##  [20,]   25  61.06640
##  [21,]  234 216.06781
##  [22,]  129 143.84279
##  [23,]  165 176.40257
##  [24,]  165  70.28128
##  [25,]   79  81.44766
##  [26,]  300 616.81963
##  [27,]  350 201.97742
##  [28,]  229 158.84502
##  [29,]   60  81.30187
##  [30,]   68 106.39305
##  [31,]  391 212.67237
##  [32,]  675 357.32024
##  [33,]  160 134.22121
##  [34,]   80 115.42298
##  [35,]   99  83.29645
##  [36,]   68 185.12896
##  [37,]  250  62.03284
##  [38,]  315 265.76639
##  [39,]   38  96.39641
##  [40,]   70  42.44675
##  [41,]  124 127.21464
##  [42,]  450 191.01387
##  [43,]  399 211.93466
##  [44,]   49 142.06827
##  [45,]  150 109.05136
##  [46,]   30  80.19047
##  [47,]  100  60.33820
```

```
##  [48,]   29  82.57054
##  [49,]  350 327.92312
##  [50,]  200 236.04542
##  [51,]   72  71.47364
##  [52,]   44  58.46617
##  [53,]   44  53.15263
##  [54,]  249 226.91976
##  [55,]  225 244.66585
##  [56,]  115 110.10549
##  [57,]  199 135.47612
##  [58,]  105  73.01431
##  [59,]  268 147.95556
##  [60,]  179  72.61688
##  [61,]  185 167.73136
##  [62,]  150  62.76781
##  [63,]  216 266.03010
##  [64,]   77  55.48656
##  [65,]  518 374.73125
##  [66,]  374 270.62392
##  [67,]  140 189.15057
##  [68,]   95  73.58303
##  [69,]  130  86.31671
##  [70,]   89  72.01955
##  [71,]   80  49.27906
##  [72,]  100  54.24685
##  [73,]  450 502.81663
##  [74,]  590 316.33620
##  [75,]  195 148.31214
##  [76,]   40  58.98364
##  [77,]   70  77.85735
##  [78,]   79  50.37094
##  [79,]   70  67.52371
##  [80,]  125 136.28016
##  [81,]   35  52.87098
##  [82,]   35  55.02415
##  [83,]   37 108.22967
##  [84,]  149  71.41112
##  [85,]   55  72.43120
##  [86,]  305 252.89541
##  [87,]  390 256.23469
##  [88,]  380 256.43408
##  [89,]  156 172.42736
##  [90,]  139 104.35807
##  [91,]  303 363.20667
##  [92,]   99 132.15096
##  [93,]  102  90.34946
##  [94,]  126  56.18512
##  [95,]   40  80.68641
##  [96,]   25  56.01167
```

```
##  [97,]  550 502.60445
##  [98,]  367 350.98740
##  [99,]  275 100.74017
## [100,]  163 225.59418
```

```
plot(pred_uk$log_data.pred,log(model_test$data), xlab="Predicted Values", ylab="Observ
ed Values")
```




```
cor(pred_uk$log_data.pred, log(model_test$data))
```

```
## [1] 0.7687945
```

```
# difference
press <- sum((log(model_test$data) - pred_uk$log_data.pred)^2)
press # PRESS for universal kriging is 238.9227
```

```
## [1] 238.9227
```

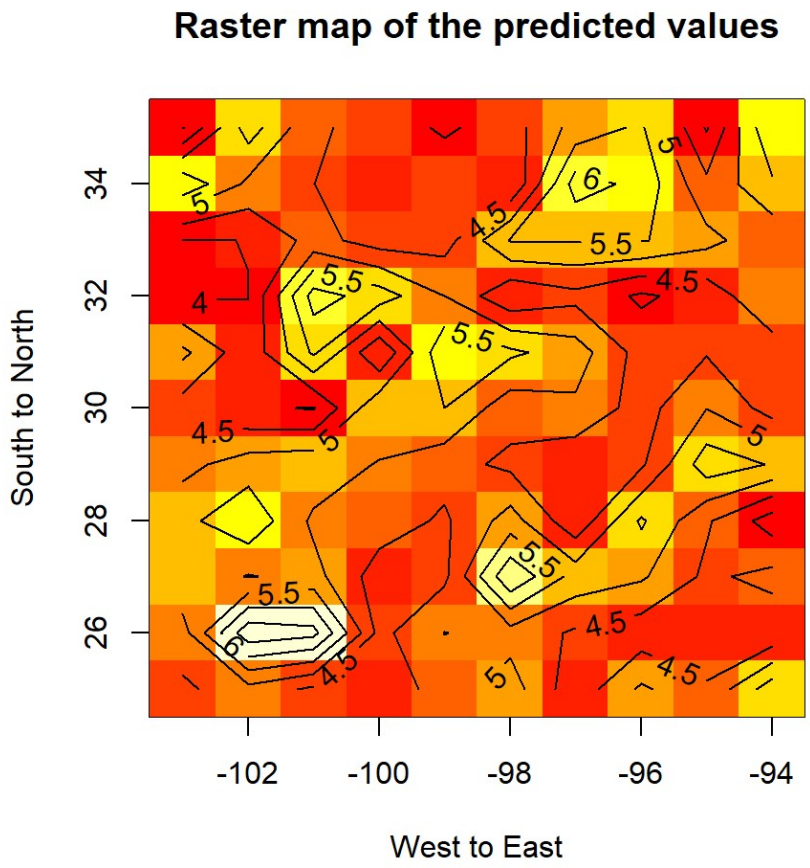
```
### Universal kriging performs much better, and the correlation jumps 20% to 0.77.

### Producing a raster map of predicted values
x.range <- as.integer(range(model_test[,1]))
y.range <- as.integer(range(model_test[,2]))

cbind(pred_uk$log_data.pred[1:10], pred_uk$log_data.var[1:10])
```

```
##           [,1]      [,2]
##  [1,] 4.394062 0.3258713
##  [2,] 4.818636 0.2951976
##  [3,] 4.389429 0.3000090
##  [4,] 4.213915 0.3033154
##  [5,] 4.693583 0.3021893
##  [6,] 5.234738 0.2974720
##  [7,] 4.179654 0.3040862
##  [8,] 5.091383 0.2948130
##  [9,] 4.580821 0.3022626
## [10,] 5.795586 0.3101480
```

```
# Raster map of predicted values
qqq <- matrix(pred_uk$log_data.pred, length(seq(from=x.range[1], to=x.range[2], by=
1)), length(seq(from=y.range[1], to=y.range[2], by=1)))
```

```
## Warning in matrix(pred_uk$log_data.pred, length(seq(from = x.range[1], to
## = x.range[2], : data length [750] is not a sub-multiple or multiple of the
## number of columns [11]
```

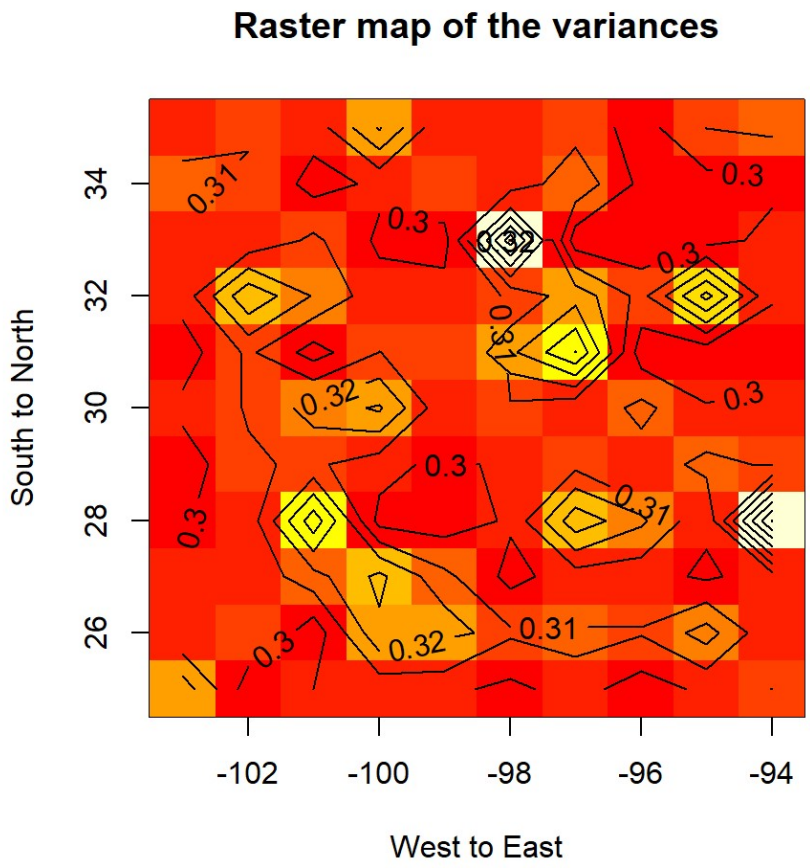
```
image(seq(from=x.range[1], to=x.range[2], by=1), seq(from=y.range[1],to=y.range[2], by
=1), qqq, xlab="West to East",ylab="South to North", main="Raster map of the predicte
d values")
contour(seq(from=x.range[1], to=x.range[2], by=1),
        seq(from=y.range[1],to=y.range[2], by=1), qqq, add=TRUE, col="black", labcex=
1)
```



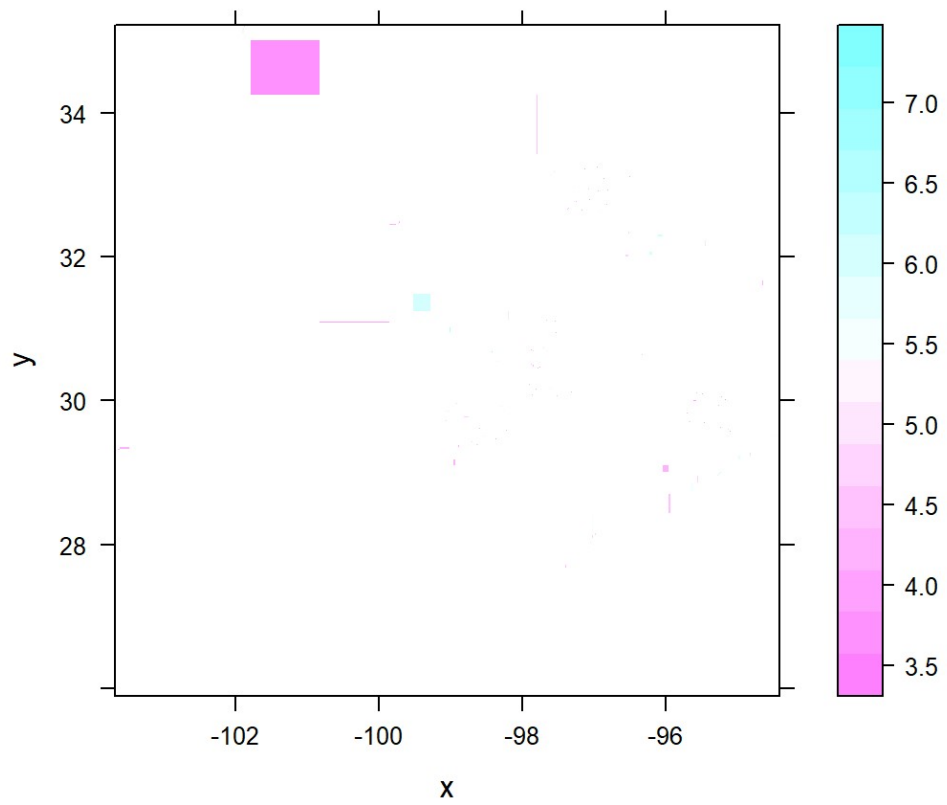

```
# Variances raster map
qqq1 <- matrix(pred_uk$log_data.var, length(seq(from=x.range[1], to=x.range[2], by=
1)), length(seq(from=y.range[1], to=y.range[2], by=1)))
```

```
## Warning in matrix(pred_uk$log_data.var, length(seq(from = x.range[1], to
## = x.range[2], : data length [750] is not a sub-multiple or multiple of the
## number of columns [11]
```

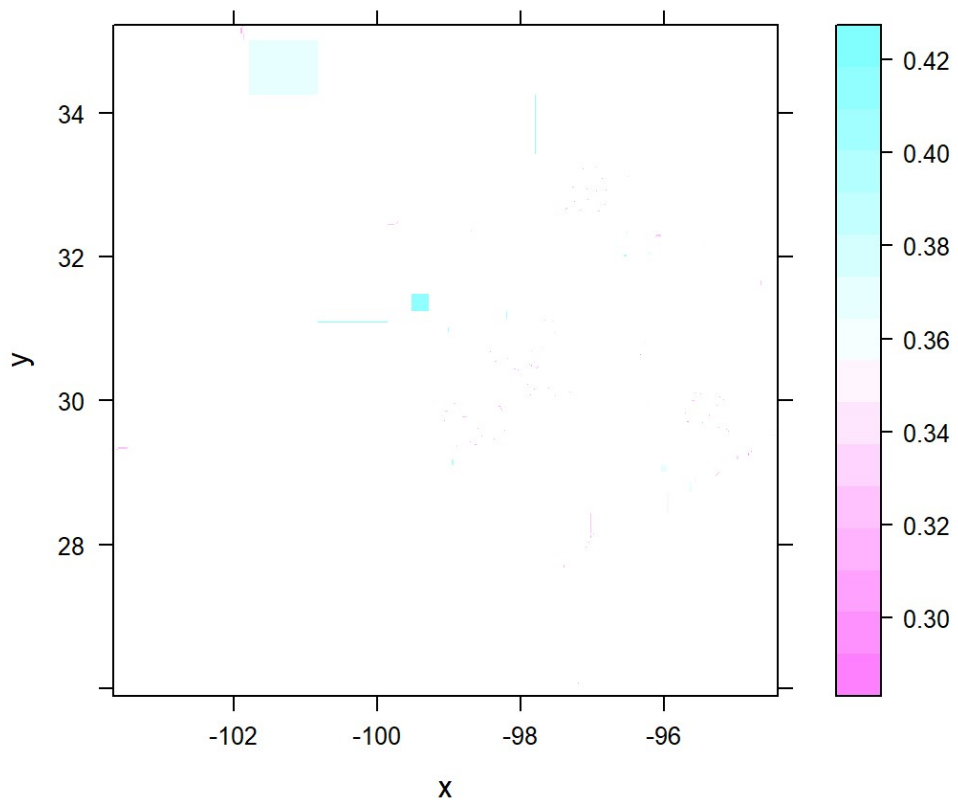
```
image(seq(from=x.range[1], to=x.range[2], by=1), seq(from=y.range[1],to=y.range[2], by
=1), qqq1, xlab="West to East",ylab="South to North", main="Raster map of the variance
s")
contour(seq(from=x.range[1], to=x.range[2], by=1),
        seq(from=y.range[1],to=y.range[2], by=1), qqq1, add=TRUE, col="black", labcex=
1)
```




```
# Using Lattice package
# A raster map using the kriged values:
levelplot(pred_uk$log_data.pred~x+y, pred_uk, aspect ="iso", main="Universal kriging p
redictions")
```




```
# A raster map using the variances of the kriged values:
levelplot(pred_uk$log_data.var~x+y, pred_uk, aspect ="iso", main="Universal kriging va
riances")
```


Universal kriging variances
34
32
30
28
y
-102
-100
-98
-96
x
0.42
0.40
0.38
0.36
0.34
0.32
0.30