C173 Project

Stephen Fujiwara

2025-03-01

EDA (Exploratory Data Analysis)

Variable Description

```
# read data
data <- read.csv("data.csv")
head(data)
```

```
##   zip_code longitude latitude population median_age percent_male percent_female
## 1    90001 -118.2468  33.9698      57110       26.6     49.84766       50.15234
## 2    90002 -118.2497  33.9511      51223       25.5     48.56412       51.43588
## 3    90003 -118.2731  33.9658      66266       26.3     49.24245       50.75755
## 4    90004 -118.3082  34.0749      62180       34.8     50.34095       49.65905
## 5    90005 -118.3097  34.0579      37681       33.9     51.21679       48.78321
## 6    90006 -118.2965  34.0471      59185       32.4     51.11768       48.88232
##   total_households avg_household_size
## 1            12971               4.40
## 2            11731               4.36
## 3            15642               4.22
## 4            22547               2.73
## 5            15044               2.50
## 6            18617               3.13
```

The dataset is a dataset from the Los Angeles county database that contains census data from each of the zip codes that *fall at least partially* within Los Angeles county boundaries, for the year 2010.

Each observation contains the zip code, the coordinates corresponding to the centroid of the zip code's area, as well as 6 variables.

These 6 variables are:

1. population - The total number of people living in a given zip code.
2. median_age - The median age of the individuals living in a given zip code.
3. percent_male - The percentage of the population that is male for a given a zip code.
4. percent_female - The percentage of the population that is female for a given zip code.
5. total_households - The total number of households located within a given zip code
6. avg_household_size - The average number of individuals living in a household located within a given zip code.

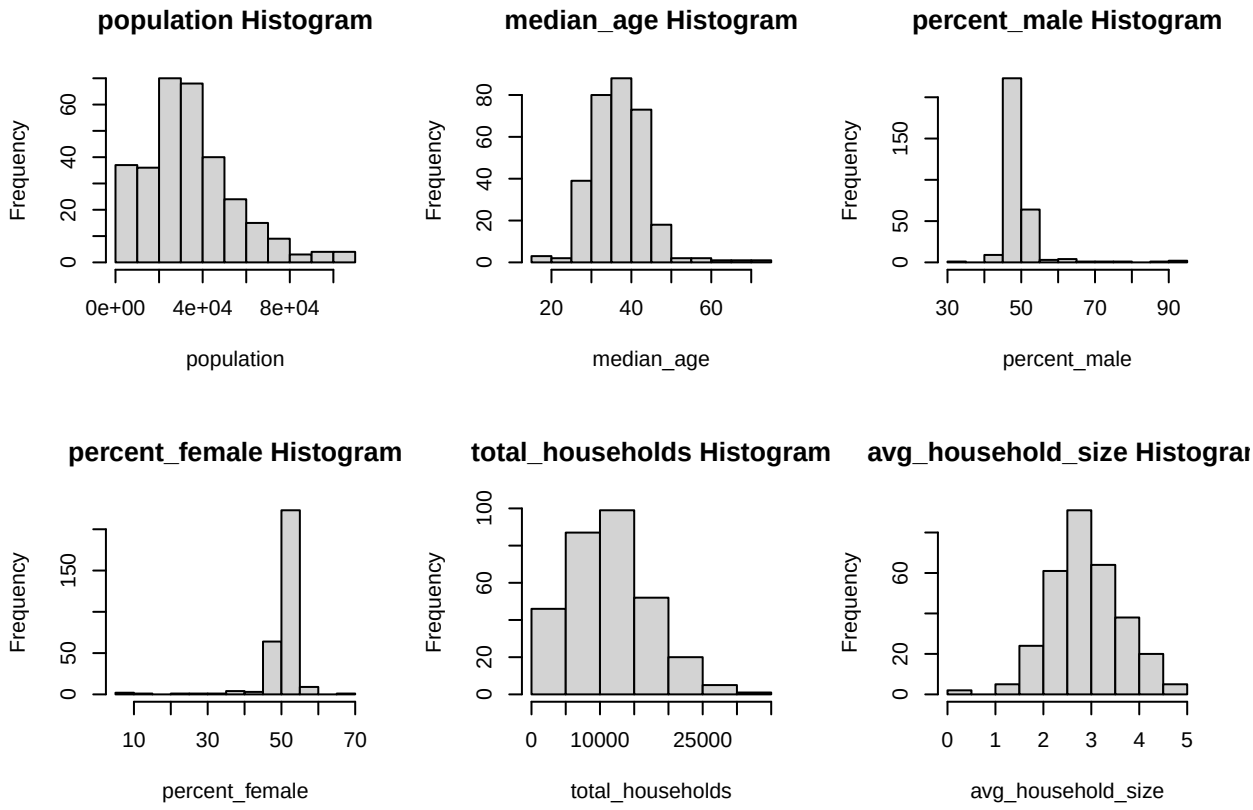
Descriptive Statistics

```r
# descriptive statistics
variables <- c("population", "median_age", "percent_male", "percent_female", "total_households", "avg_h

summary(data[, variables])
```

```
##    population       median_age     percent_male   percent_female
##  Min.   :    15   Min.   :19.30   Min.   :32.69   Min.   : 6.122
##  1st Qu.: 20401   1st Qu.:32.60   1st Qu.:48.20   1st Qu.:50.004
##  Median : 31922   Median :37.20   Median :48.88   Median :51.117
##  Mean   : 34193   Mean   :37.06   Mean   :49.79   Mean   :50.210
##  3rd Qu.: 45129   3rd Qu.:41.00   3rd Qu.:50.00   3rd Qu.:51.803
##  Max.   :105549   Max.   :74.00   Max.   :93.88   Max.   :67.308
##  total_households avg_household_size
##  Min.   :    0    Min.   :0.000
##  1st Qu.: 7307    1st Qu.:2.450
##  Median :11204    Median :2.865
##  Mean   :11278    Mean   :2.894
##  3rd Qu.:15013    3rd Qu.:3.330
##  Max.   :31087    Max.   :4.670
```

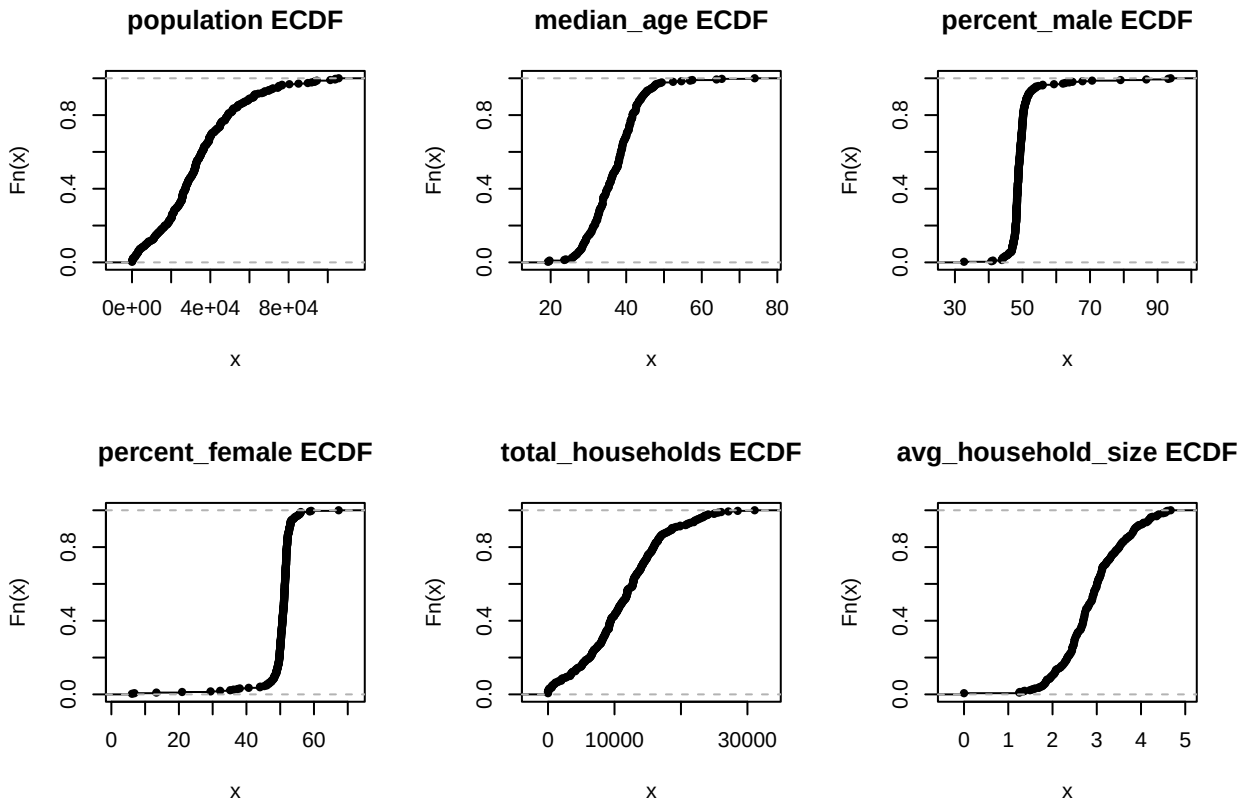
Histogram of Each Variable

```r
# histograms
par(mfrow = c(2, length(variables) / 2))
for (variable in variables) {
  hist(data[, variable], main = paste(variable, "Histogram"), xlab = variable)
}
```



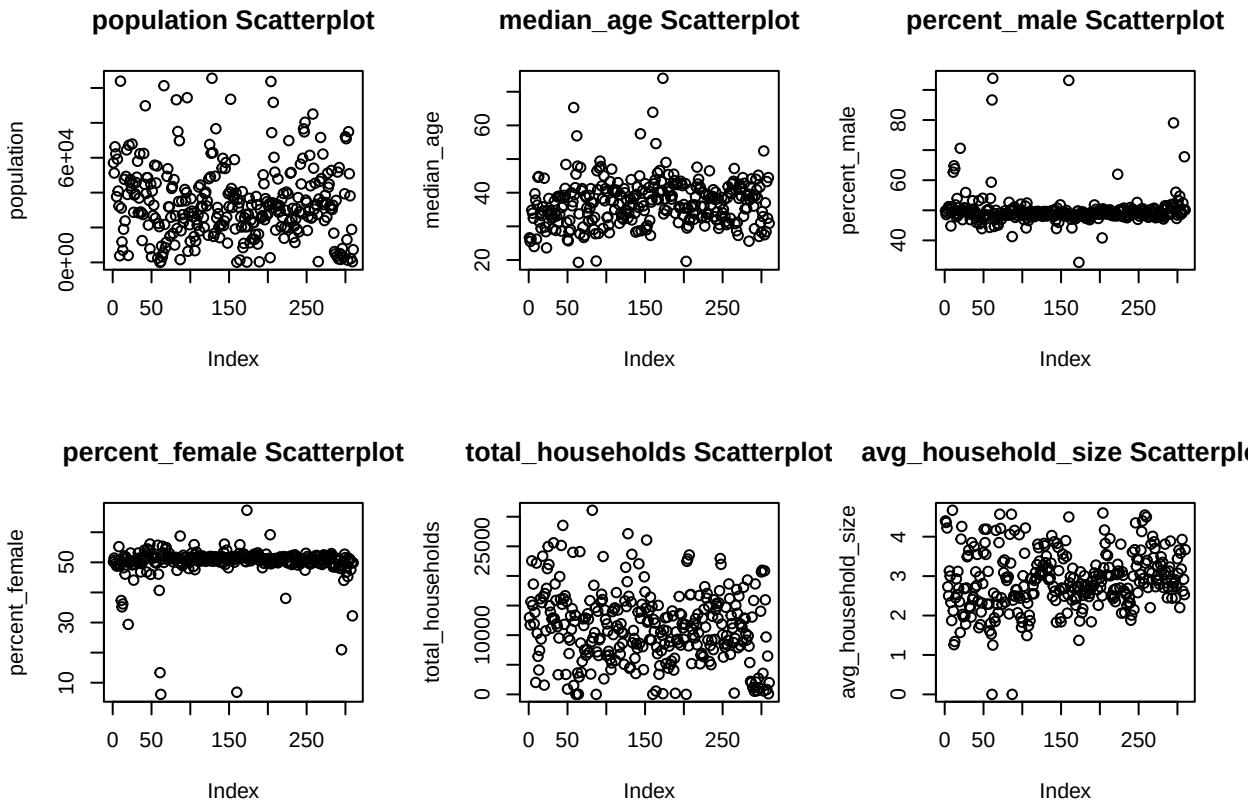
ECDF of Each Variable

```
# ECDFs
par(mfrow = c(2, length(variables) / 2))
for (variable in variables) {
  plot(ecdf(data[, variable]), main = paste(variable, "ECDF"))
}
```



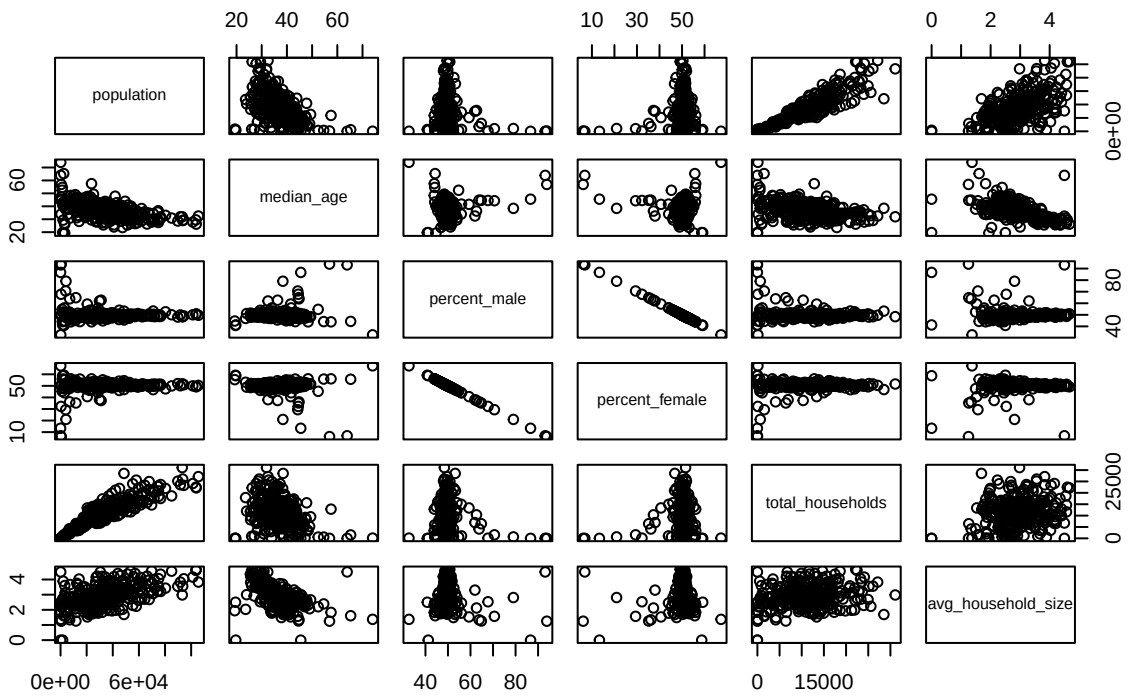
Scatterplot of Each Variable (Individual)

```
# scatterplots
par(mfrow = c(2, length(variables) / 2))
for (variable in variables) {
  plot(data[, variable], main = paste(variable, "Scatterplot"), ylab = variable)
}
```



Scatterplot of Each Variable (Pair-wise)

```
# scatterplots (pair-wise)
pairs(data[, variables], main = "Pair-wise Scatter Plots")
```

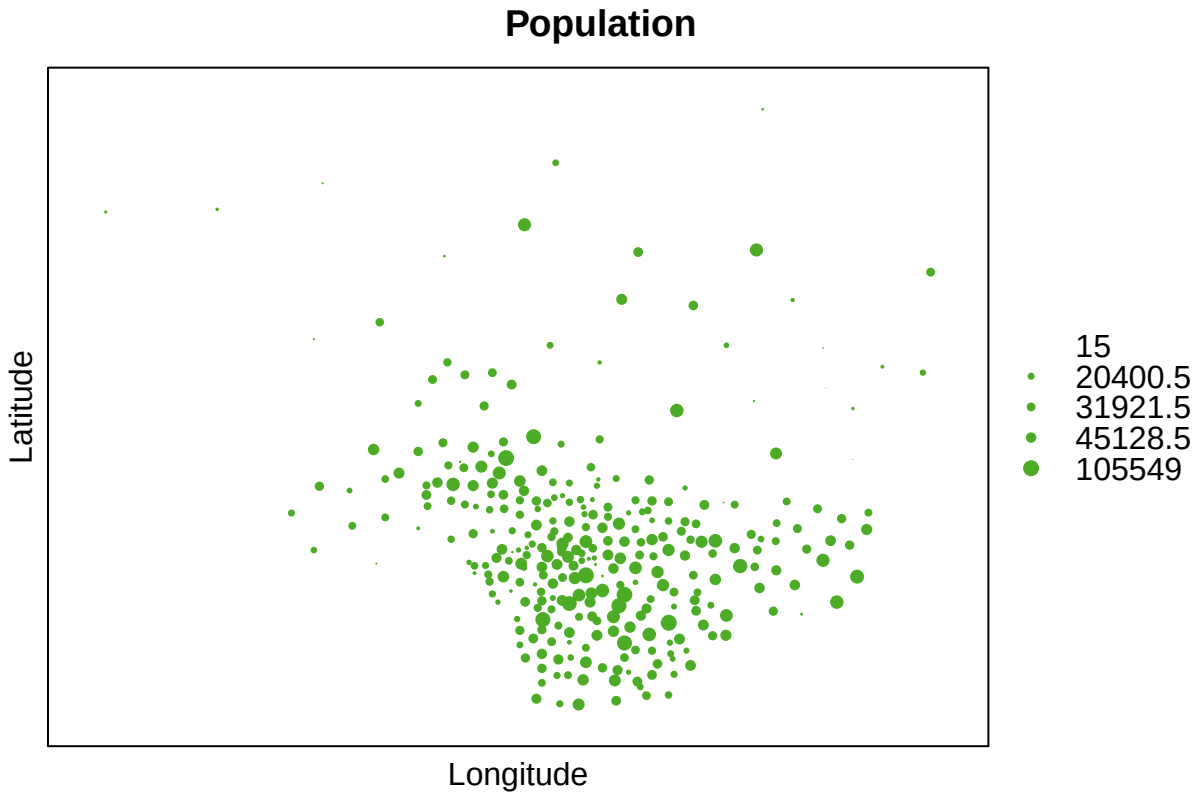


Bubble Plot of Each Variable

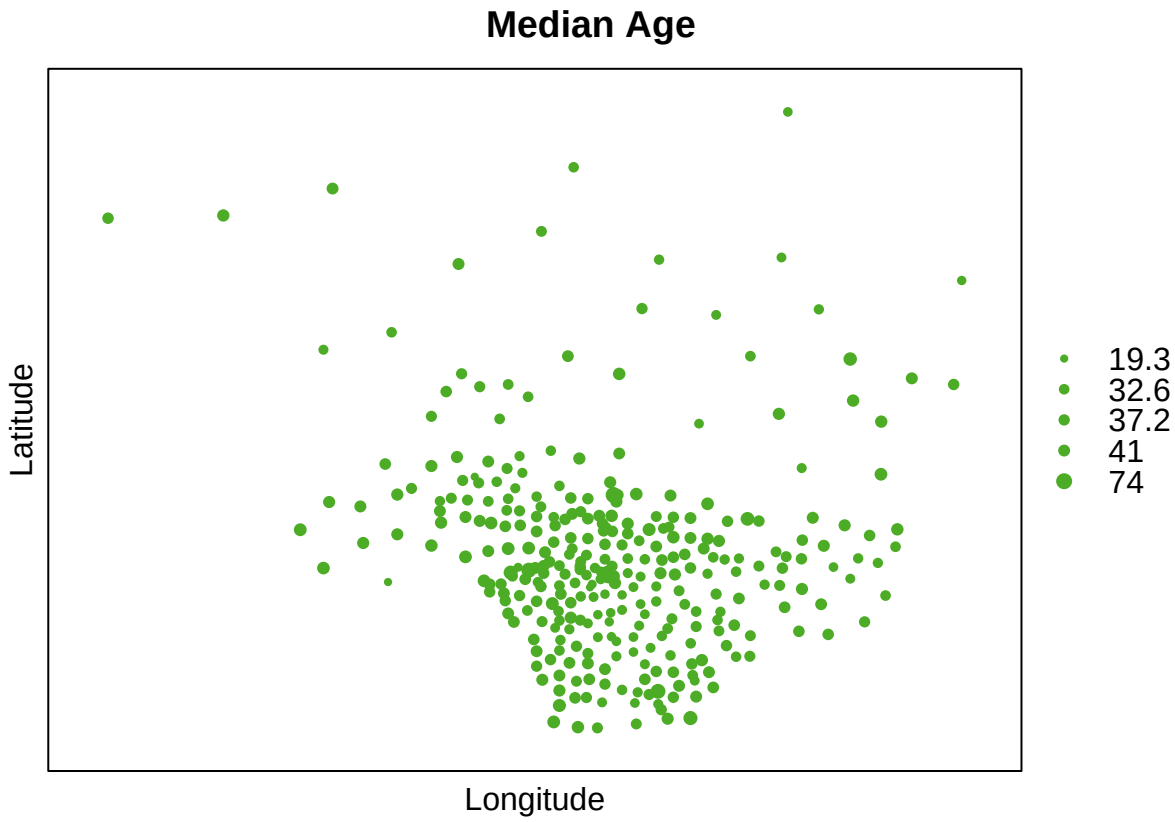
```
library(sp)
```

```
## Warning: package 'sp' was built under R version 4.4.2
```

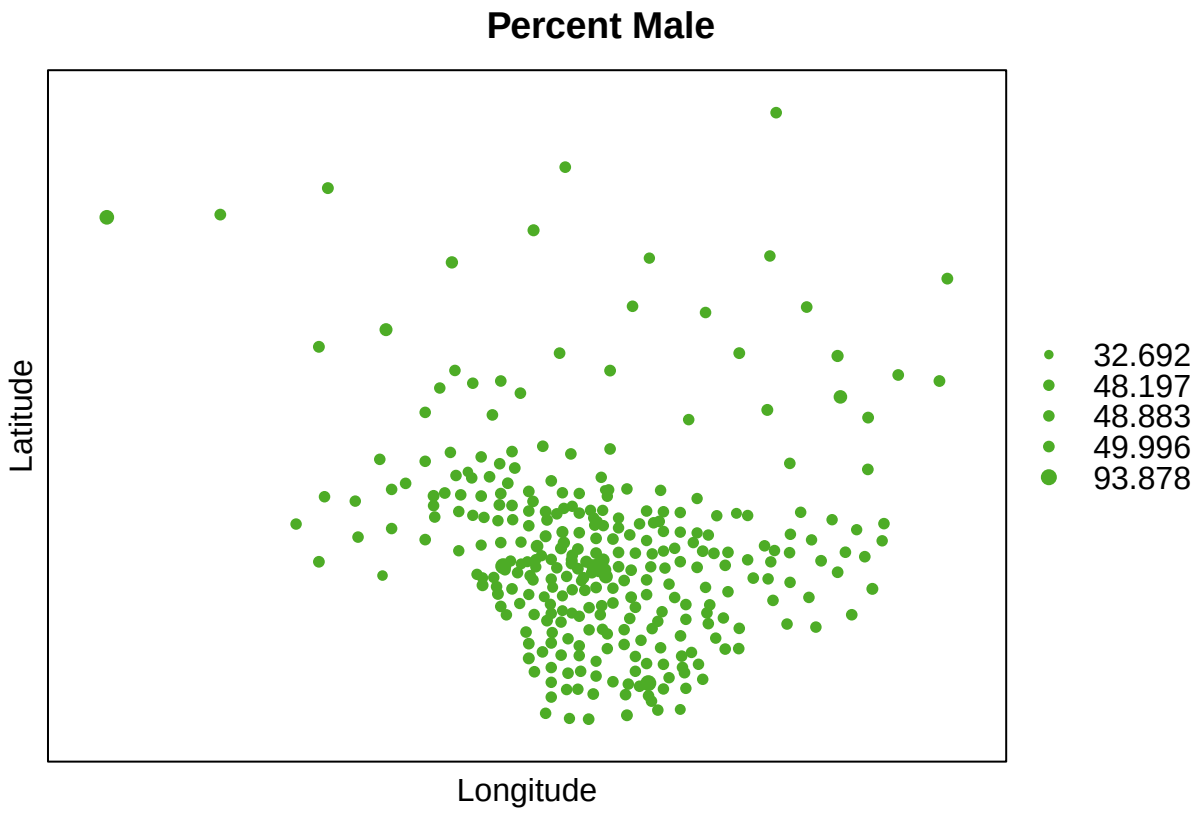
```
# bubble plots of each variable

coordinates(data) <- ~ longitude + latitude

bubble(data, "population", maxsize = 1, main = "Population", xlab = "Longitude", ylab = "Latitude")
```



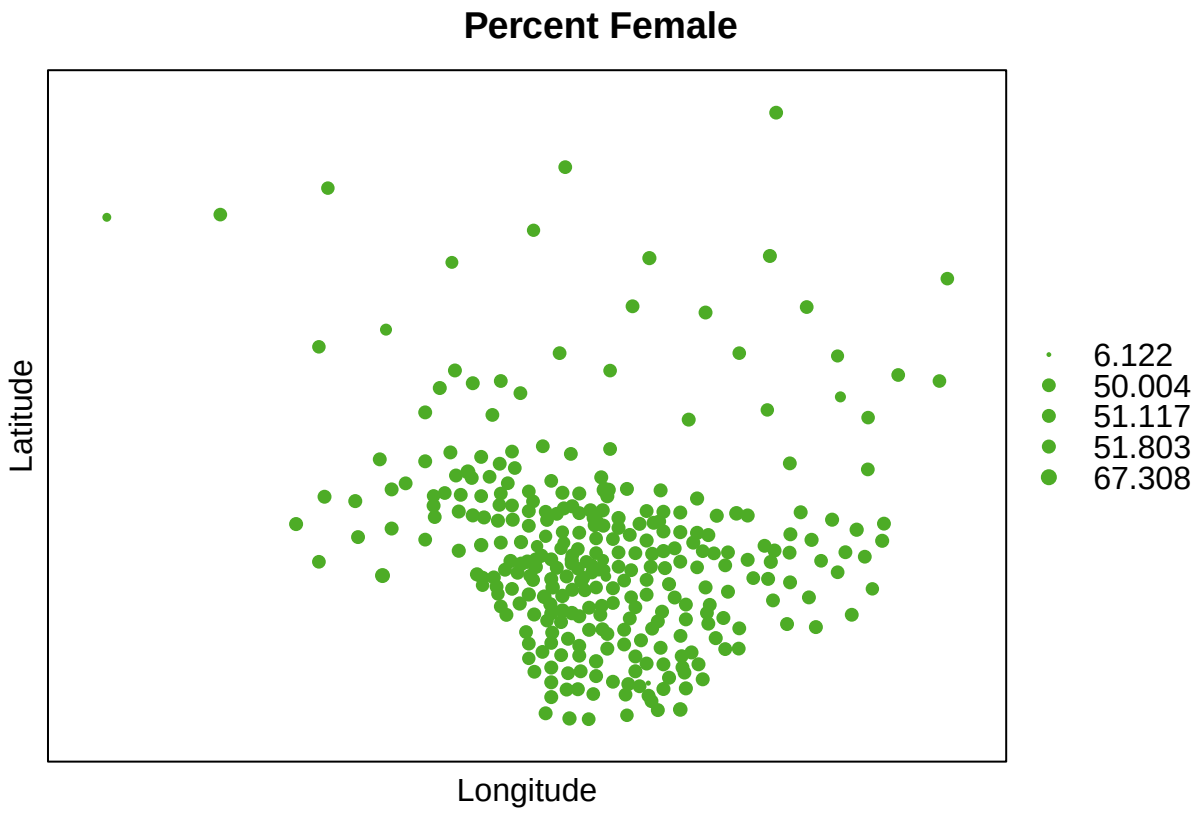

```
bubble(data, "median_age", maxsize = 1, main = "Median Age", xlab = "Longitude", ylab = "Latitude")
```



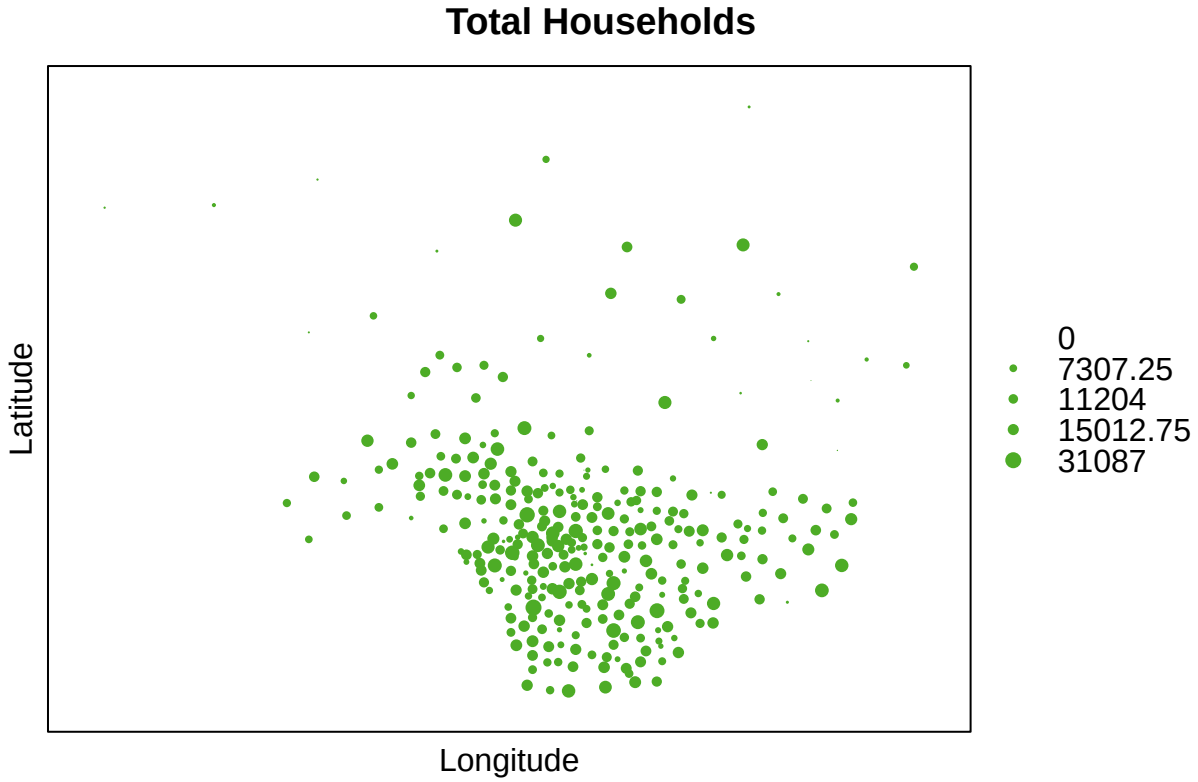

```
bubble(data, "percent_male", maxsize = 1, main = "Percent Male", xlab = "Longitude", ylab = "Latitude")
```



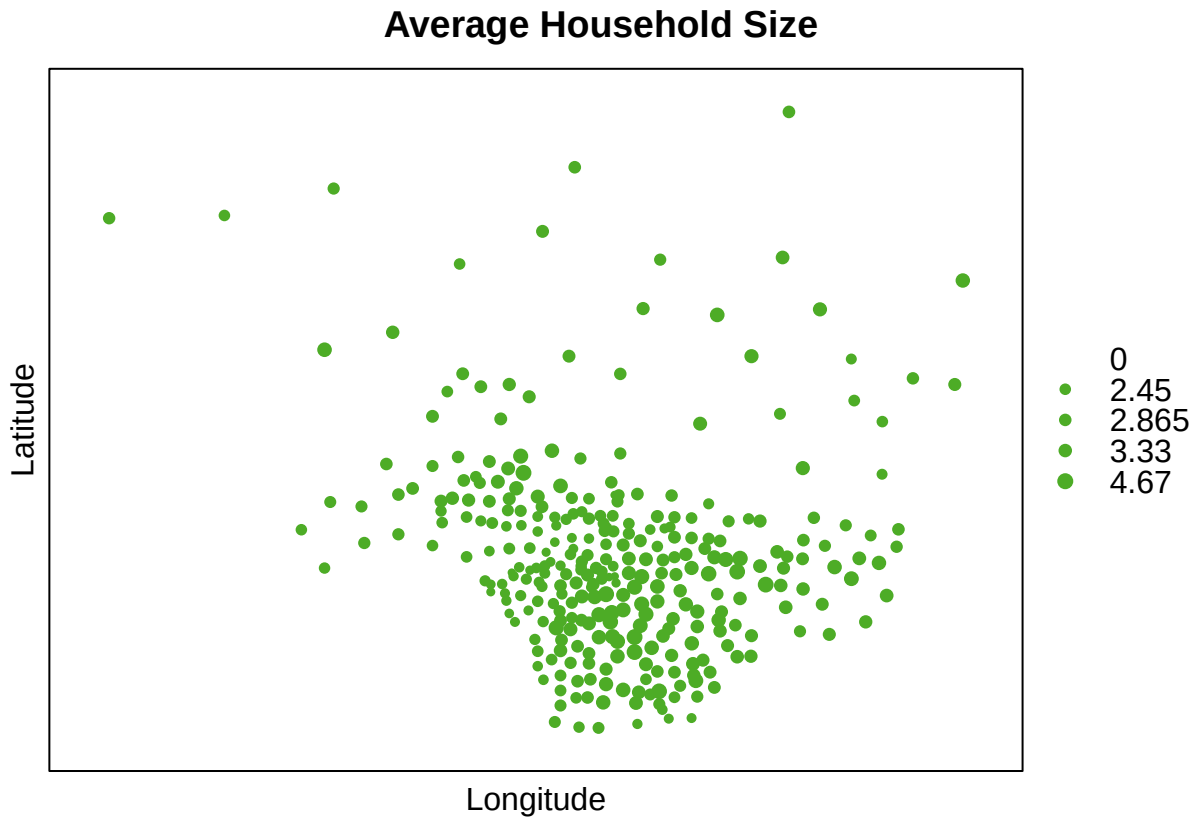

```
bubble(data, "percent_female", maxsize = 1, main = "Percent Female", xlab = "Longitude", ylab = "Latitu
```




```
bubble(data, "total_households", maxsize = 1, main = "Total Households", xlab = "Longitude", ylab = "La
```




```
bubble(data, "avg_household_size", maxsize = 1, main = "Average Household Size", xlab = "Longitude", yla
```




```
# revert to regular dataframe
data <- as.data.frame(data)
```

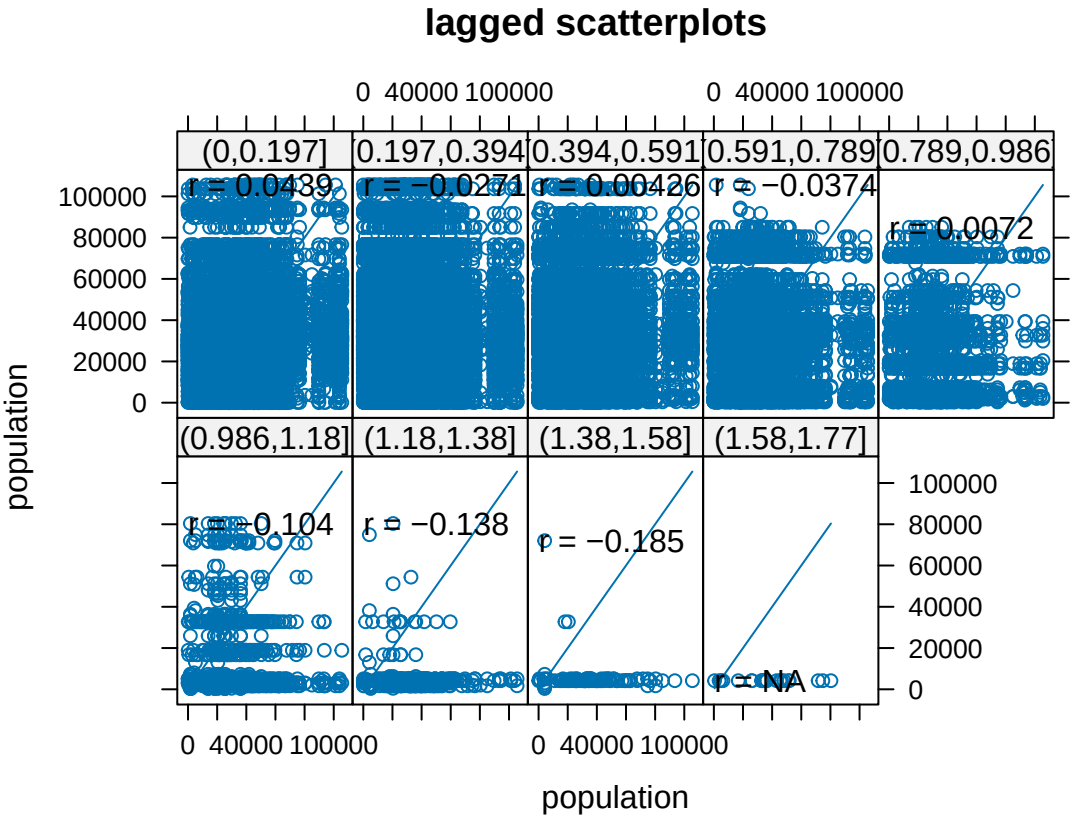
H-scatterplot of Each Variable

```
library(gstat)
```

```
## Warning: package 'gstat' was built under R version 4.4.2
```

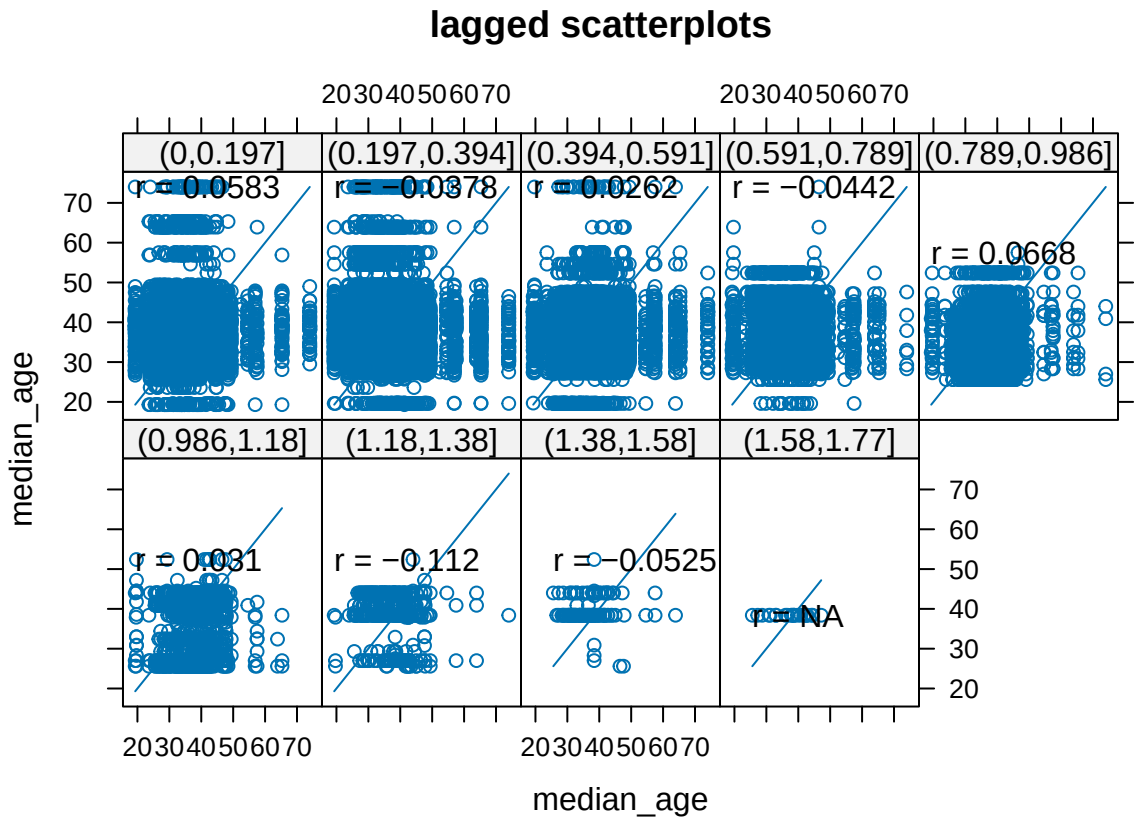
```
# h-scatter plots of each variable

# convert to spatial dataframe object
coordinates(data) <-  ~ longitude + latitude

# compute max distance between coordinates of the dataset
max_distance <- max(dist(coordinates(data)))

# compute sequence of h-values to plot with
h <- seq(0, max_distance, length.out = 10)

hscat(population ~ 1, data, h)
```

```
## Warning in cor(x, y): the standard deviation is zero
```



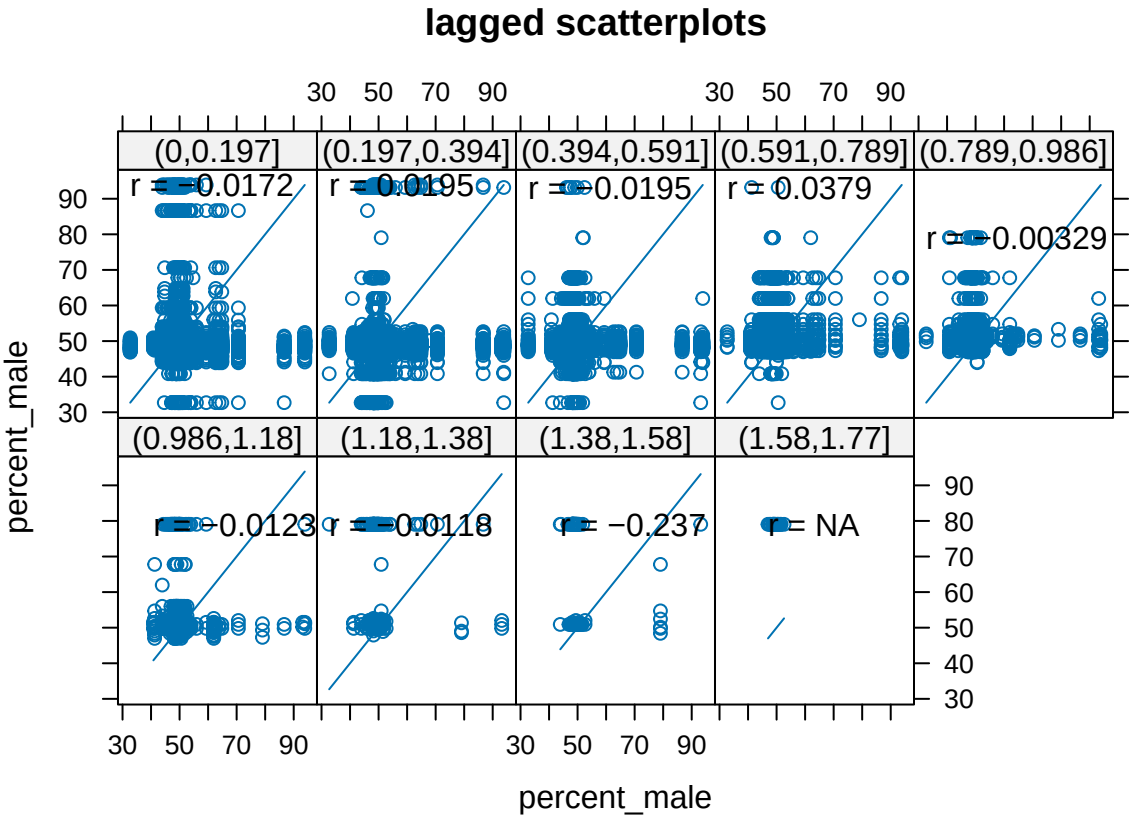

```
hscat(median_age ~ 1, data, h)
```

```
## Warning in cor(x, y): the standard deviation is zero
```



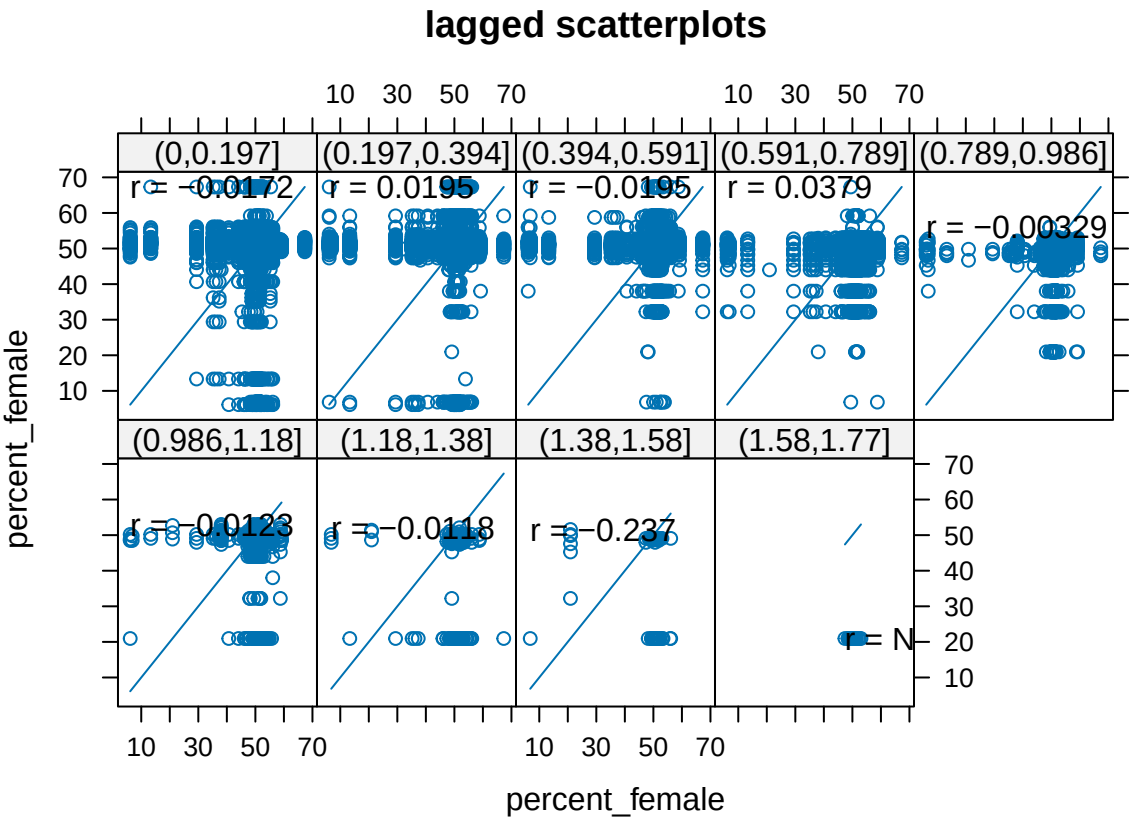

```
hscat(percent_male ~ 1, data, h)
```

```
## Warning in cor(x, y): the standard deviation is zero
```



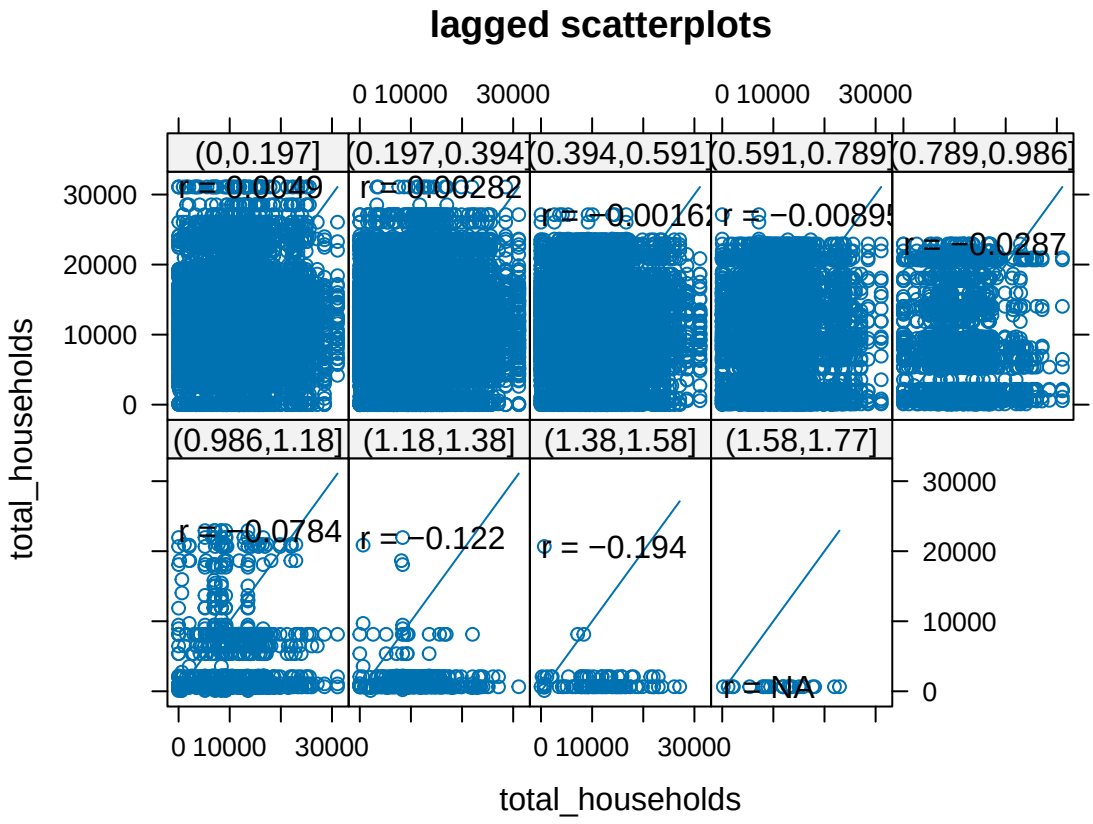

```
hscat(percent_female ~ 1, data, h)
```

```
## Warning in cor(x, y): the standard deviation is zero
```



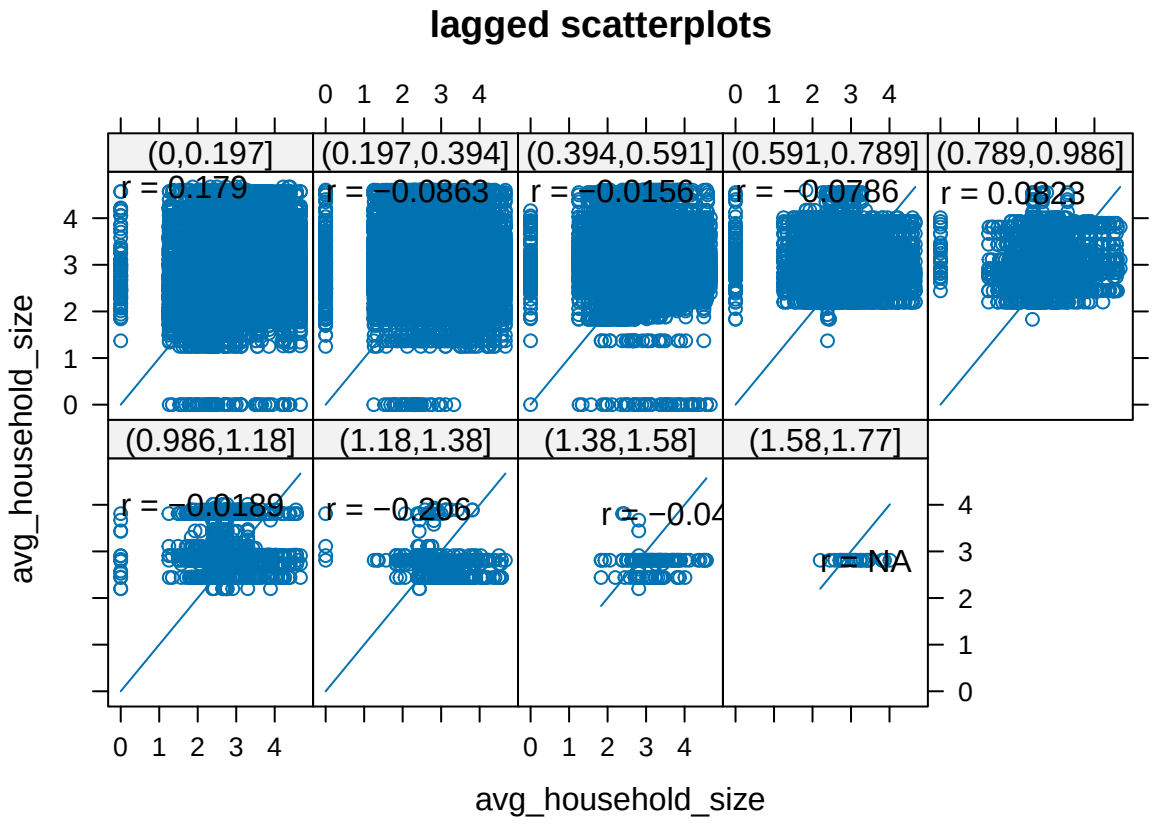

```
hscat(total_households ~ 1, data, h)
```

```
## Warning in cor(x, y): the standard deviation is zero
```




```
hscat(avg_household_size ~ 1, data, h)
```

```
## Warning in cor(x, y): the standard deviation is zero
```




```
# convert back to regular dataframe
data <- as.data.frame(data)
```

Variogram Modeling

Fitting a Model to the Empirical Semivariogram (Classical)

```
library(geoR)
```

Computing the Empirical Semivariogram (Classical)

```
## Warning: package 'geoR' was built under R version 4.4.2

## ----------------------------------------------------------
##  Analysis of Geostatistical Data
##  For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
##  geoR version 1.9-4 (built on 2024-02-14) is now loaded
## ----------------------------------------------------------
```

```
# create dataframe of just coordinates and target variable
target_variable <- "population"
target_data <- data[, c("longitude", "latitude", target_variable)]
head(target_data)
```

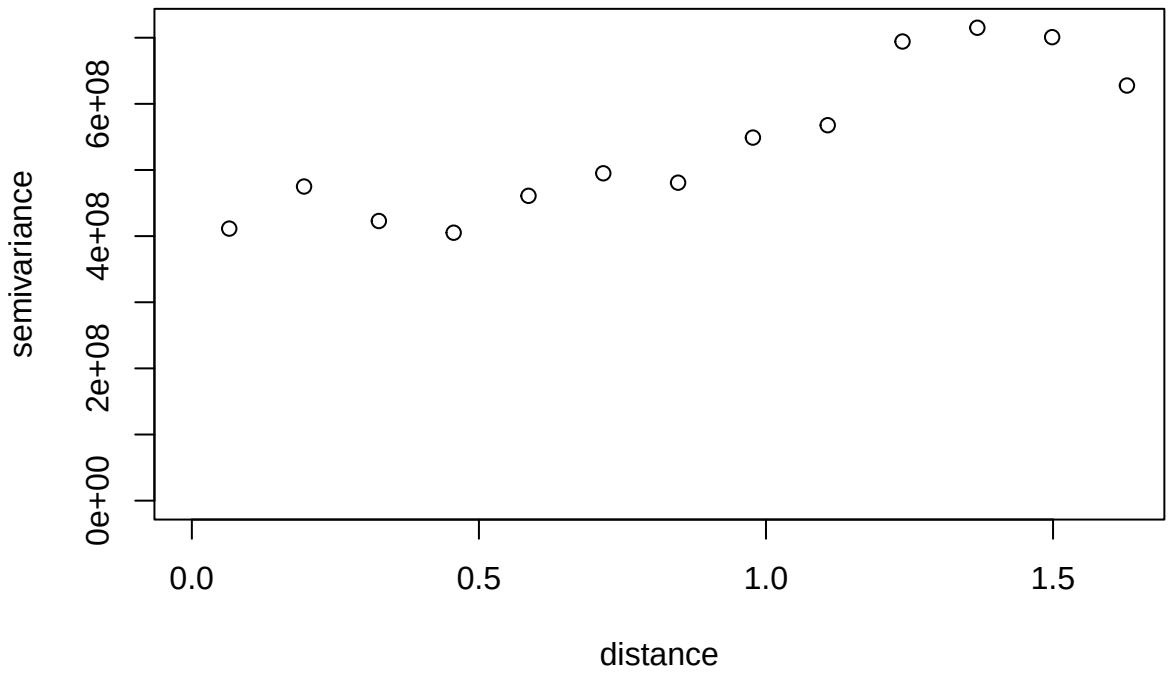
```
##   longitude latitude population
## 1 -118.2468  33.9698      57110
## 2 -118.2497  33.9511      51223
## 3 -118.2731  33.9658      66266
## 4 -118.3082  34.0749      62180
## 5 -118.3097  34.0579      37681
## 6 -118.2965  34.0471      59185
```

```r
# set target variable and create geodata object
geo_data <- as.geodata(target_data)
```

```r
# compute empirical semivariogram (classical)
empirical_semivariogram_c <- variog(geo_data, max.dist = 1.7)
```

```
## variog: computing omnidirectional variogram
```

```r
plot(empirical_semivariogram_c)
```




```r
# fitting a model to the empirical semivariogram (classical) - spherical
plot(empirical_semivariogram_c)
```

```r
cov_model <- "sph"
cov_pars <- c(3*10^8, 1.4)
nugget <- 4*10^8

# fitting by eye
lines.variomodel(cov.model = cov_model, cov.pars = cov_pars, nugget = nugget)

# fitting using equal weights
equal_weights_model <- variofit(empirical_semivariogram_c, cov.model = cov_model, ini.cov.pars = cov_pa
```

Fitting a Spherical Model

```
## variofit: covariance model used is spherical
## variofit: weights used: equal
## variofit: minimisation function used: optim
```

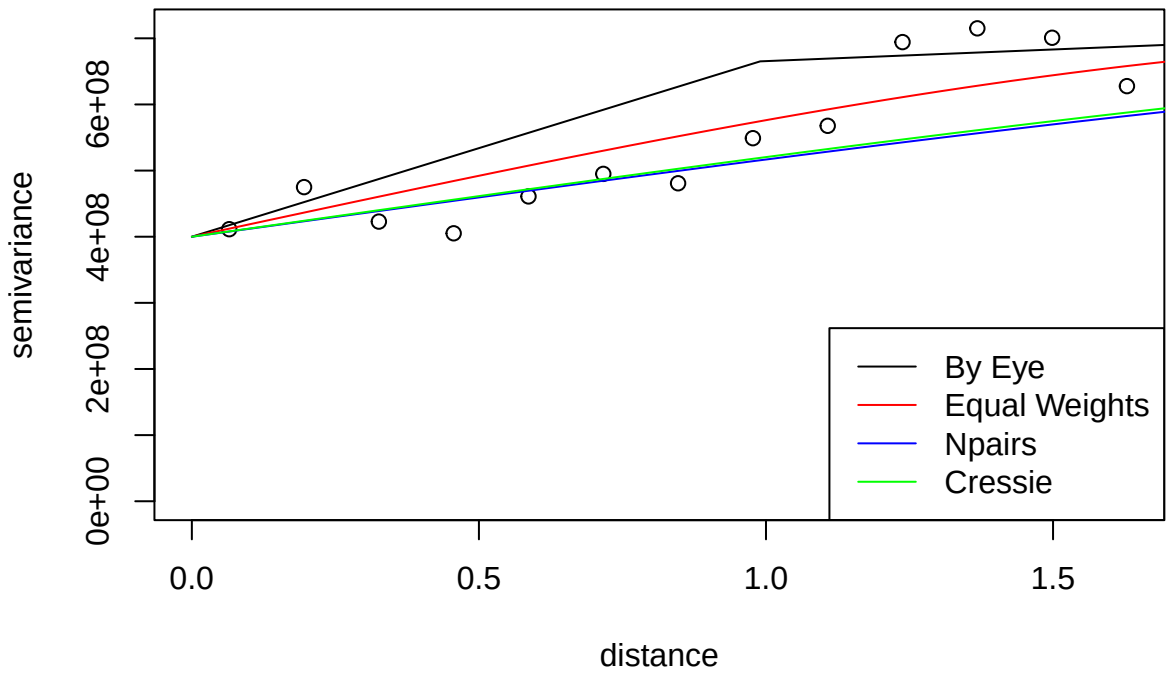
```r
lines(equal_weights_model, lty = 1, col = "red")

# fitting using npairs weights
npairs_weights_model <- variofit(empirical_semivariogram_c, cov.model = cov_model, ini.cov.pars = cov_p
```

```
## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```r
lines(npairs_weights_model, lty = 1, col = "blue")

# fitting using cressie's weights
cressies_weights_model <- variofit(empirical_semivariogram_c, cov.model = cov_model, ini.cov.pars = cov
```

```
## variofit: covariance model used is spherical
## variofit: weights used: cressie
## variofit: minimisation function used: optim
```

```r
lines(cressies_weights_model, lty = 1, col = "green")

# legend
legend(
  "bottomright",
  legend = c("By Eye", "Equal Weights", "Npairs", "Cressie"),
  col = c("black", "red", "blue", "green"),
  lty = 1
)
```



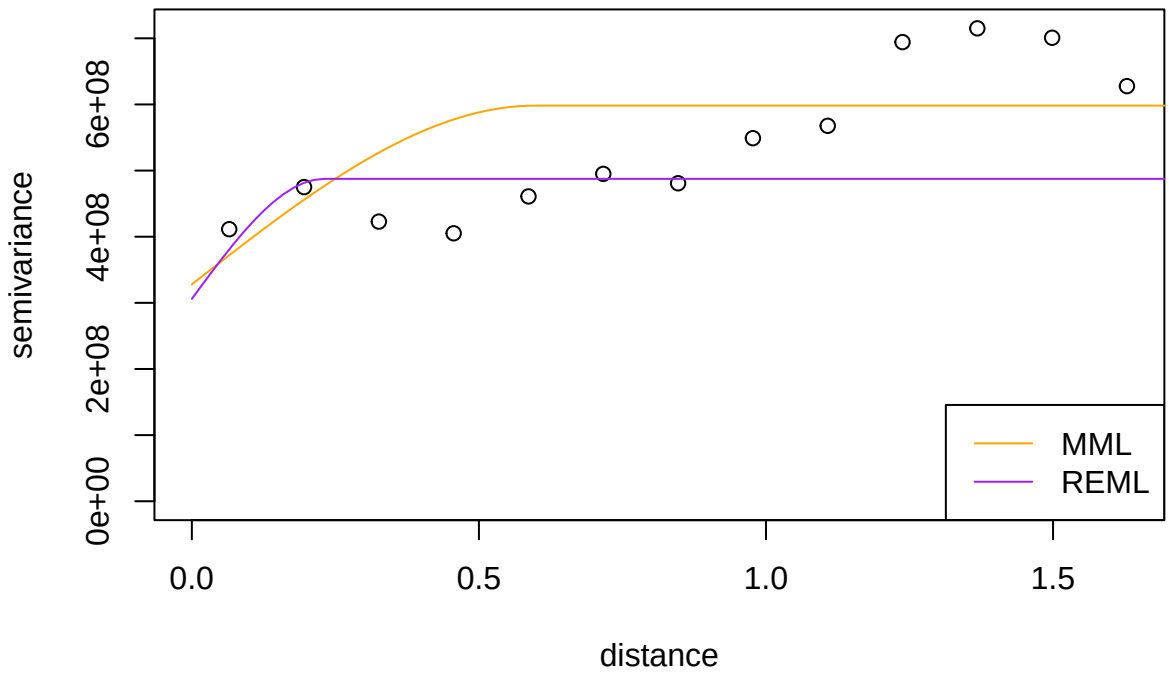

```
# fitting a model using MLE - spherical
# although the empirical semivariogram is not used for fitting, we would like to observe it's fit to th
plot(empirical_semivariogram_c)

cov_model <- "sph"
cov_pars <- c(3*10^8, 1.4)
nugget <- 4*10^8

# fitting using MML
mml_model <- likfit(geo_data, cov.model = cov_model, ini.cov.pars = cov_pars,
       fix.nugget = FALSE, nugget = nugget)
```

```
## kappa not used for the spherical correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
lines(mml_model, lty = 1, col="orange")

# fitting using REML
reml_model <- likfit(geo_data, cov.model = cov_model, ini.cov.pars = cov_pars,
       fix.nugget = FALSE, nugget = nugget, lik.method = "RML")
```

```
## kappa not used for the spherical correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
lines(reml_model, lty = 1, col = "purple")

# legend
legend(
  "bottomright",
  legend = c("MML", "REML"),
  col = c("orange", "purple"),
  lty = 1
)
```




```
# fitting a model to the empirical semivariogram (classical) - exponential
plot(empirical_semivariogram_c)

cov_model <- "exp"
cov_pars <- c(3*10^8, 1.4)
nugget <- 4*10^8

# fitting by eye
lines.variomodel(cov.model = cov_model, cov.pars = cov_pars, nugget = nugget)

# fitting using equal weights
equal_weights_model <- variofit(empirical_semivariogram_c, cov.model = cov_model, ini.cov.pars = cov_pa
```

Fitting an Exponential Model

```
## variofit: covariance model used is exponential
## variofit: weights used: equal
## variofit: minimisation function used: optim
```

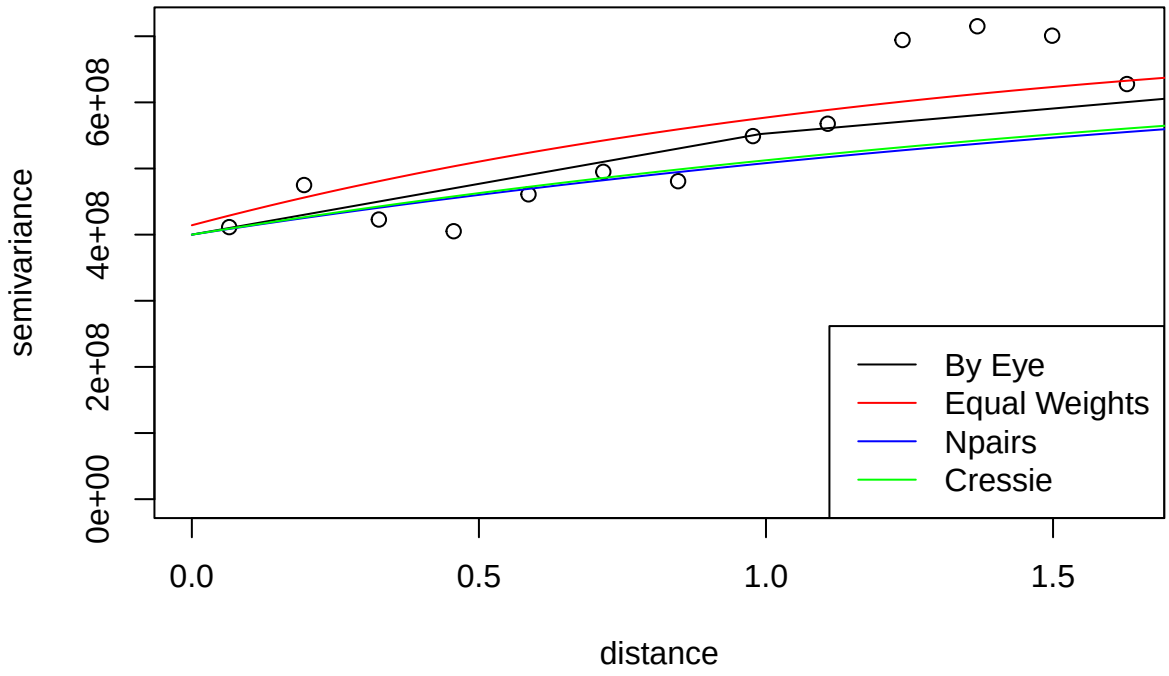
```
lines(equal_weights_model, lty = 1, col = "red")

# fitting using npairs weights
npairs_weights_model <- variofit(empirical_semivariogram_c, cov.model = cov_model, ini.cov.pars = cov_pa
```

```
## variofit: covariance model used is exponential
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
lines(npairs_weights_model, lty = 1, col = "blue")

# fitting using cressie's weights
cressies_weights_model <- variofit(empirical_semivariogram_c, cov.model = cov_model, ini.cov.pars = cov
```

```
## variofit: covariance model used is exponential
## variofit: weights used: cressie
## variofit: minimisation function used: optim
```

```
lines(cressies_weights_model, lty = 1, col = "green")

# legend
legend(
  "bottomright",
  legend = c("By Eye", "Equal Weights", "Npairs", "Cressie"),
  col = c("black", "red", "blue", "green"),
  lty = 1
)
```



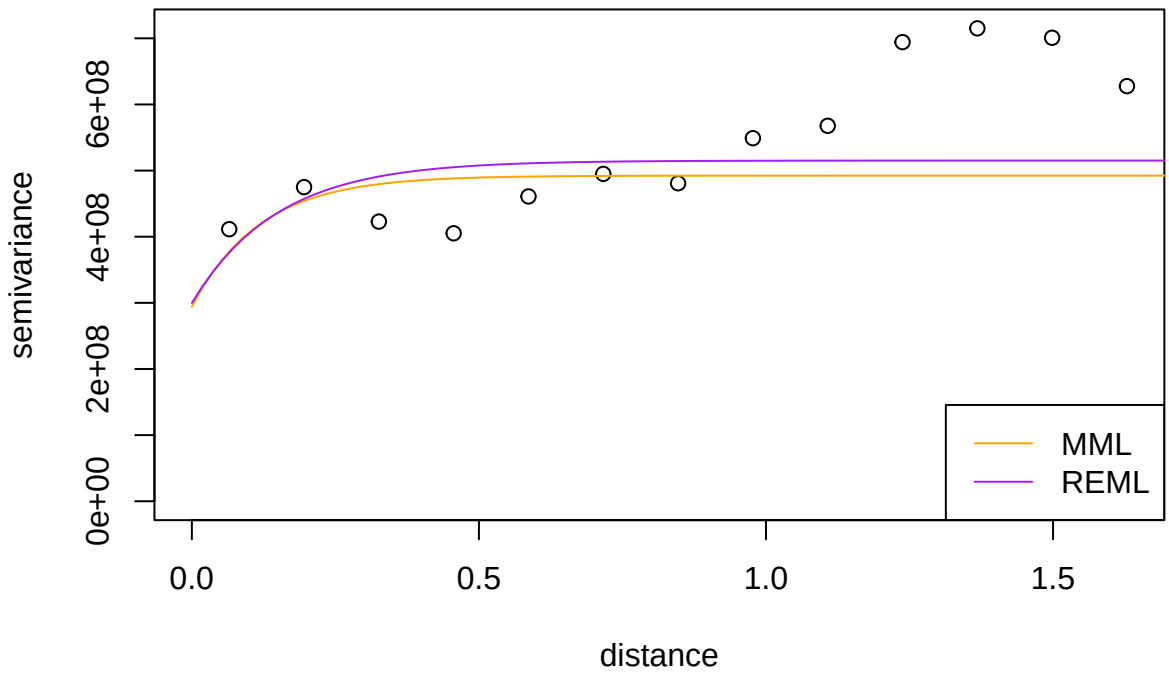
```r
# fitting a model using MLE - exponential
# although the empirical semivariogram is not used for fitting, we would like to observe it's fit to th
plot(empirical_semivariogram_c)

cov_model <- "exp"
cov_pars <- c(3*10^8, 1.4)
nugget <- 4*10^8

# fitting using MML
mml_model <- likfit(geo_data, cov.model = cov_model, ini.cov.pars = cov_pars,
       fix.nugget = FALSE, nugget = nugget)
```

```
## kappa not used for the exponential correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
lines(mml_model, lty = 1, col="orange")

# fitting using REML
reml_model <- likfit(geo_data, cov.model = cov_model, ini.cov.pars = cov_pars,
       fix.nugget = FALSE, nugget = nugget, lik.method = "RML")
```

```
## kappa not used for the exponential correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
lines(reml_model, lty = 1, col = "purple")

# legend
legend(
  "bottomright",
  legend = c("MML", "REML"),
  col = c("orange", "purple"),
  lty = 1
)
```




```
# fitting a model to the empirical semivariogram (classical) - gaussian
plot(empirical_semivariogram_c)

cov_model <- "gauss"
cov_pars <- c(3*10^8, 1.4)
nugget <- 4*10^8

# fitting by eye
lines.variomodel(cov.model = cov_model, cov.pars = cov_pars, nugget = nugget)

# fitting using equal weights
equal_weights_model <- variofit(empirical_semivariogram_c, cov.model = cov_model, ini.cov.pars = cov_pa
```

Fitting a Gaussian Model

```
## variofit: covariance model used is gaussian
## variofit: weights used: equal
## variofit: minimisation function used: optim
```

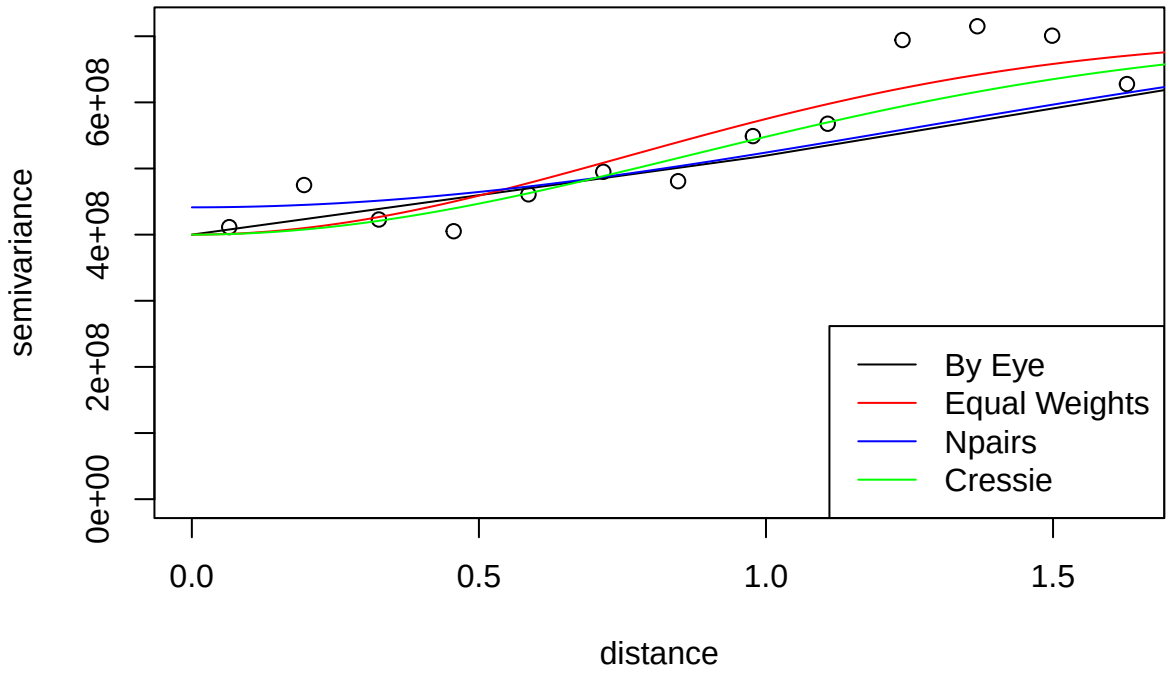
```
lines(equal_weights_model, lty = 1, col = "red")

# fitting using npairs weights
npairs_weights_model <- variofit(empirical_semivariogram_c, cov.model = cov_model, ini.cov.pars = cov_pa
```

```
## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
lines(npairs_weights_model, lty = 1, col = "blue")

# fitting using cressie's weights
cressies_weights_model <- variofit(empirical_semivariogram_c, cov.model = cov_model, ini.cov.pars = cov
```

```
## variofit: covariance model used is gaussian
## variofit: weights used: cressie
## variofit: minimisation function used: optim
```

```
lines(cressies_weights_model, lty = 1, col = "green")

# legend
legend(
  "bottomright",
  legend = c("By Eye", "Equal Weights", "Npairs", "Cressie"),
  col = c("black", "red", "blue", "green"),
  lty = 1
)
```



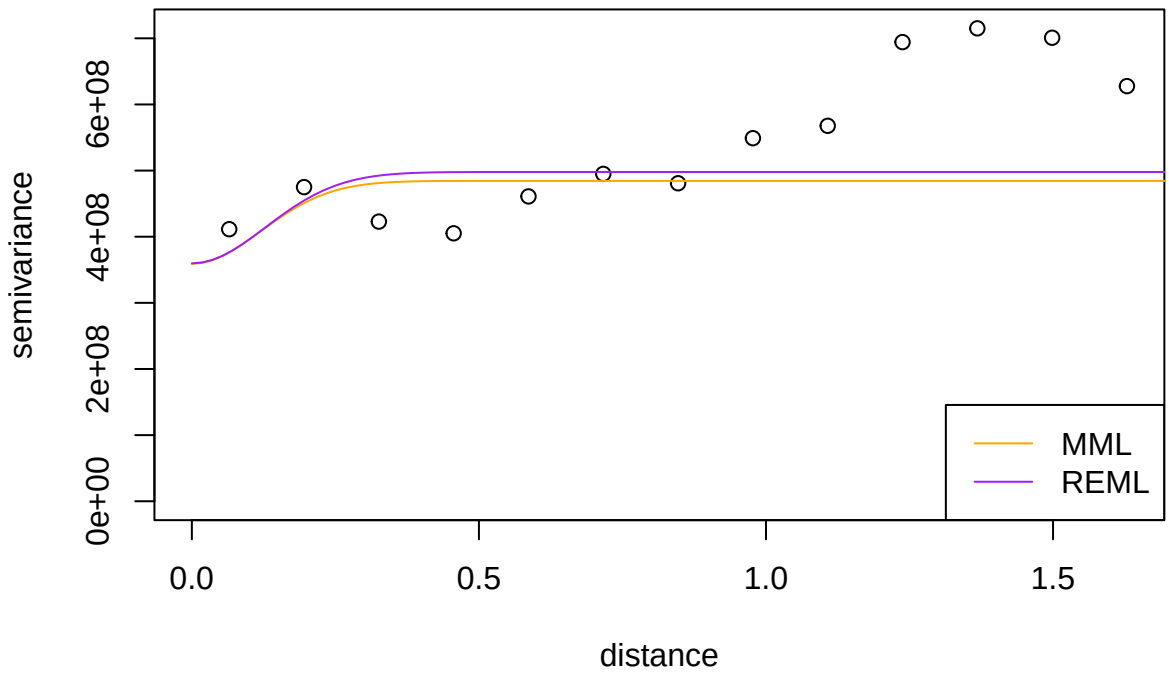
```r
# fitting a model using MLE - gaussian
# although the empirical semivariogram is not used for fitting, we would like to observe it's fit to th
plot(empirical_semivariogram_c)

cov_model <- "gauss"
cov_pars <- c(3*10^8, 1.4)
nugget <- 4*10^8

# fitting using MML
mml_model <- likfit(geo_data, cov.model = cov_model, ini.cov.pars = cov_pars,
       fix.nugget = FALSE, nugget = nugget)
```

```
## kappa not used for the gaussian correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
lines(mml_model, lty = 1, col="orange")

# fitting using REML
reml_model <- likfit(geo_data, cov.model = cov_model, ini.cov.pars = cov_pars,
       fix.nugget = FALSE, nugget = nugget, lik.method = "RML")
```

```
## kappa not used for the gaussian correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
lines(reml_model, lty = 1, col = "purple")

# legend
legend(
  "bottomright",
  legend = c("MML", "REML"),
  col = c("orange", "purple"),
  lty = 1
)
```



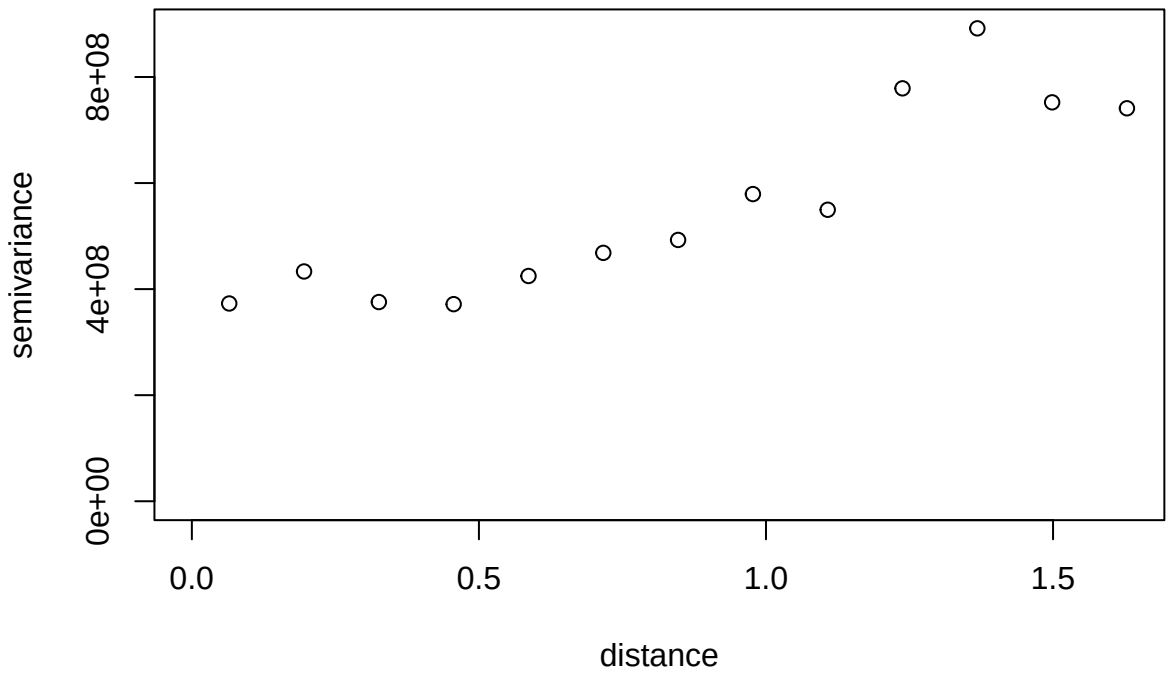
Fitting a Model to the Empirical Semivariogram (Robust)

```
# compute empirical semivariogram (robust)
empirical_semivariogram_r <- variog(geo_data, estimator.type = "modulus" , max.dist = 1.7)
```

Computing the Empirical Semivariogram (Robust)

```
## variog: computing omnidirectional variogram
```

```
plot(empirical_semivariogram_r)
```




```
# fitting a model to the empirical semivariogram (robust) - spherical
plot(empirical_semivariogram_r)

cov_model <- "sph"
cov_pars <- c(4*10^8, 1.4)
nugget <- 4*10^8

# fitting by eye
lines.variomodel(cov.model = cov_model, cov.pars = cov_pars, nugget = nugget)

# fitting using equal weights
equal_weights_model <- variofit(empirical_semivariogram_r, cov.model = cov_model, ini.cov.pars = cov_pa
```

Fitting a Spherical Model

```
## variofit: covariance model used is spherical
## variofit: weights used: equal
## variofit: minimisation function used: optim
```

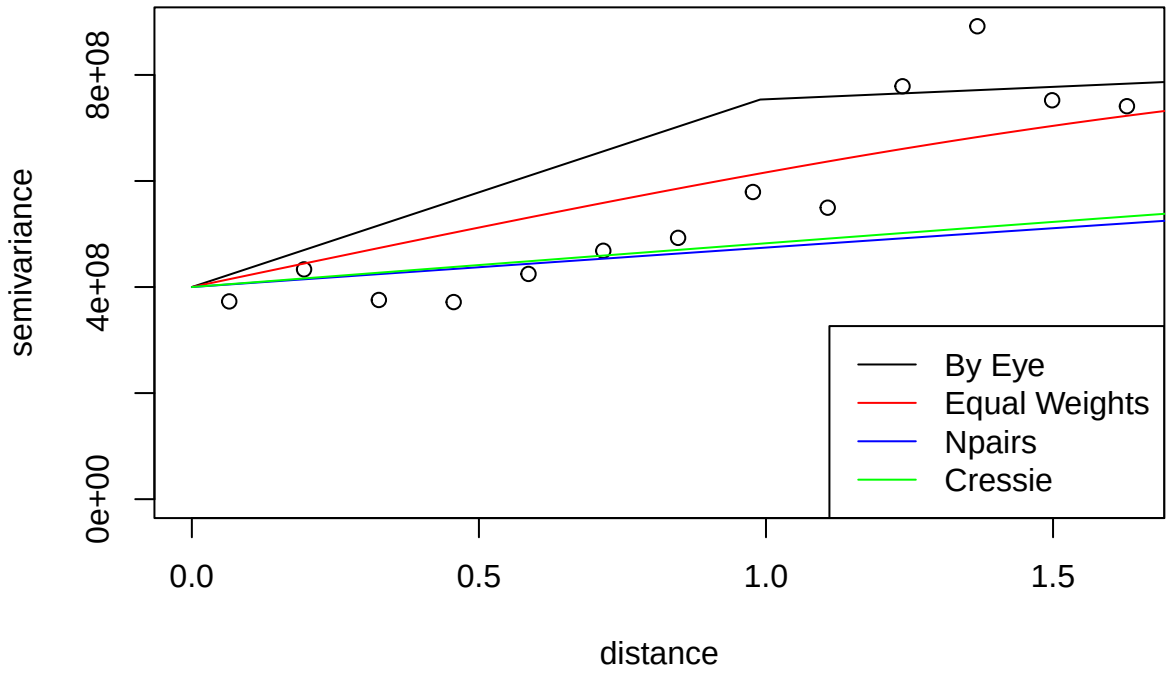
```
lines(equal_weights_model, lty = 1, col = "red")

# fitting using npairs weights
npairs_weights_model <- variofit(empirical_semivariogram_r, cov.model = cov_model, ini.cov.pars = cov_pa
```

```
## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
lines(npairs_weights_model, lty = 1, col = "blue")

# fitting using cressie's weights
cressies_weights_model <- variofit(empirical_semivariogram_r, cov.model = cov_model, ini.cov.pars = cov
```

```
## variofit: covariance model used is spherical
## variofit: weights used: cressie
## variofit: minimisation function used: optim
```

```
lines(cressies_weights_model, lty = 1, col = "green")

# legend
legend(
  "bottomright",
  legend = c("By Eye", "Equal Weights", "Npairs", "Cressie"),
  col = c("black", "red", "blue", "green"),
  lty = 1
)
```



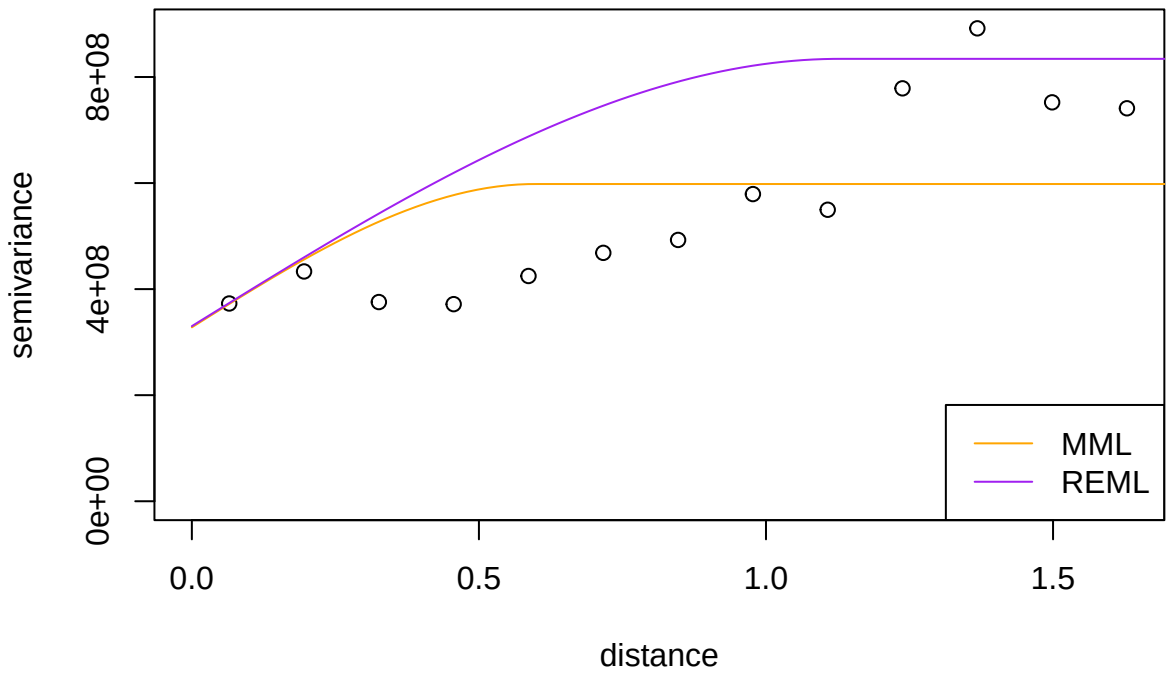
```r
# fitting a model using MLE - spherical
# although the empirical semivariogram is not used for fitting, we would like to observe it's fit to th

plot(empirical_semivariogram_r)

cov_model <- "sph"
cov_pars <- c(4*10^8, 1.4)
nugget <- 4*10^8

# fitting using MML
mml_model <- likfit(geo_data, cov.model = cov_model, ini.cov.pars = cov_pars,
       fix.nugget = FALSE, nugget = nugget)
```

```
## kappa not used for the spherical correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
lines(mml_model, lty = 1, col="orange")

# fitting using REML
reml_model <- likfit(geo_data, cov.model = cov_model, ini.cov.pars = cov_pars,
       fix.nugget = FALSE, nugget = nugget, lik.method = "RML")
```

```
## kappa not used for the spherical correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
lines(reml_model, lty = 1, col = "purple")

# legend
legend(
  "bottomright",
  legend = c("MML", "REML"),
  col = c("orange", "purple"),
  lty = 1
)
```




```
# fitting a model to the empirical semivariogram (robust) - exponential
plot(empirical_semivariogram_r)

cov_model <- "exp"
cov_pars <- c(4*10^8, 1.4)
nugget <- 4*10^8

# fitting by eye
lines.variomodel(cov.model = cov_model, cov.pars = cov_pars, nugget = nugget)

# fitting using equal weights
equal_weights_model <- variofit(empirical_semivariogram_r, cov.model = cov_model, ini.cov.pars = cov_pa
```

Fitting an Exponential Model

```
## variofit: covariance model used is exponential
## variofit: weights used: equal
## variofit: minimisation function used: optim
```

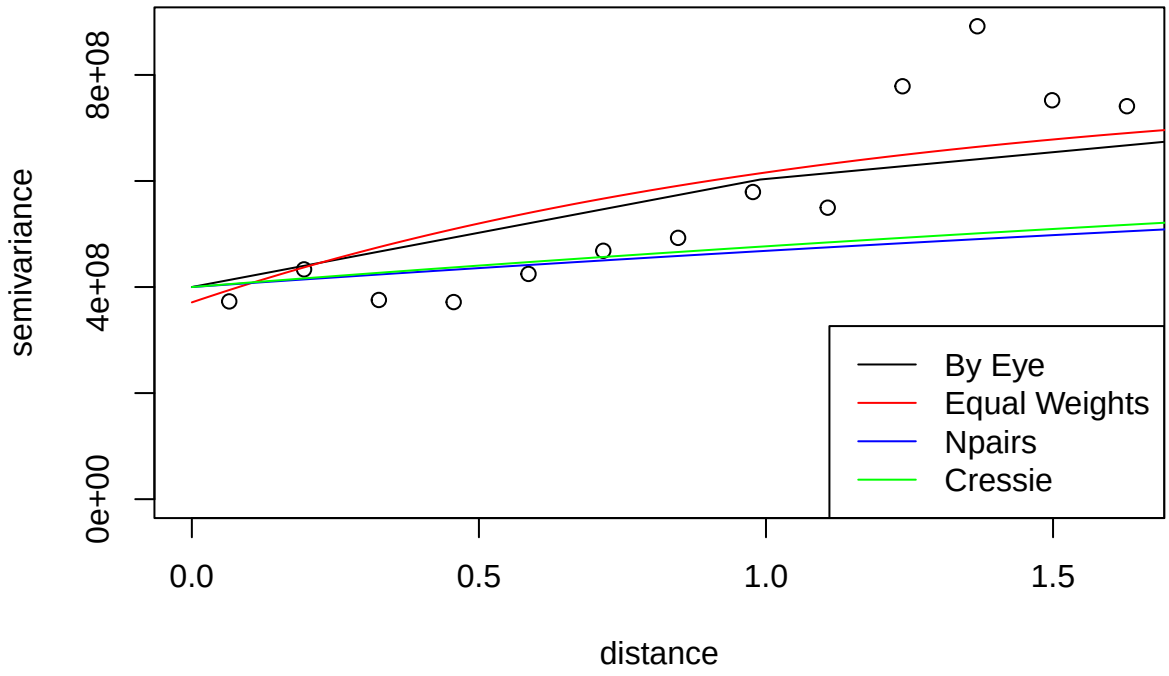
```
lines(equal_weights_model, lty = 1, col = "red")

# fitting using npairs weights
npairs_weights_model <- variofit(empirical_semivariogram_r, cov.model = cov_model, ini.cov.pars = cov_pa
```

```
## variofit: covariance model used is exponential
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
lines(npairs_weights_model, lty = 1, col = "blue")

# fitting using cressie's weights
cressies_weights_model <- variofit(empirical_semivariogram_r, cov.model = cov_model, ini.cov.pars = cov
```

```
## variofit: covariance model used is exponential
## variofit: weights used: cressie
## variofit: minimisation function used: optim
```

```
lines(cressies_weights_model, lty = 1, col = "green")

# legend
legend(
  "bottomright",
  legend = c("By Eye", "Equal Weights", "Npairs", "Cressie"),
  col = c("black", "red", "blue", "green"),
  lty = 1
)
```



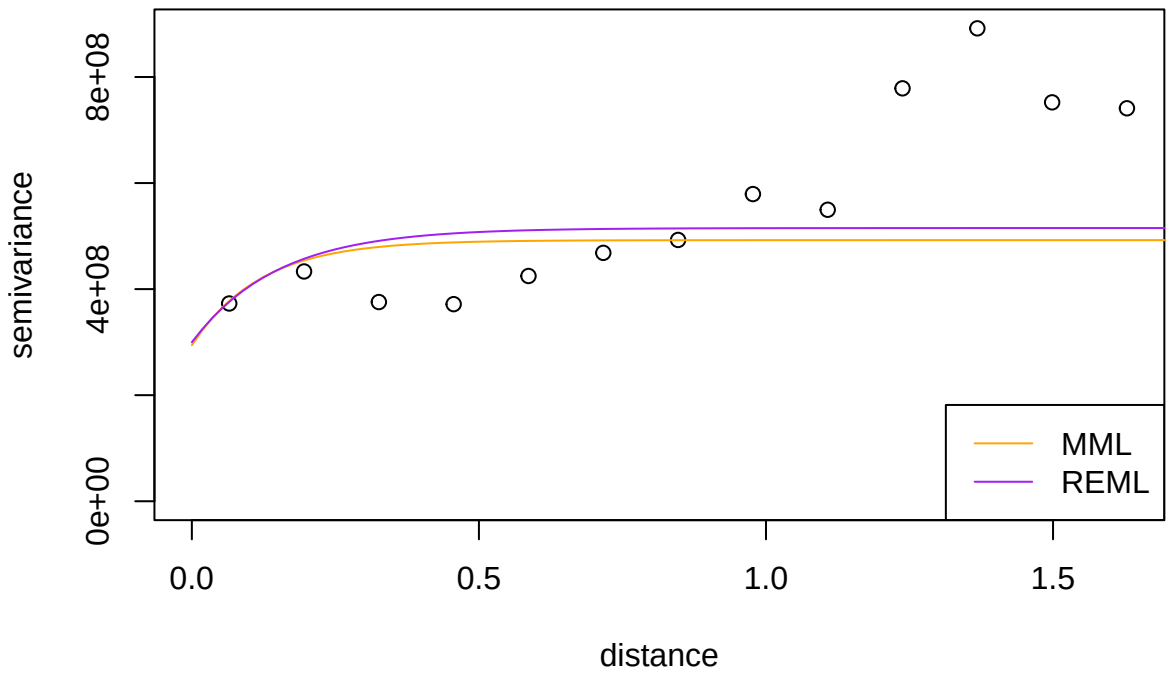
```r
# fitting a model using MLE - exponential
# although the empirical semivariogram is not used for fitting, we would like to observe it's fit to th

plot(empirical_semivariogram_r)

cov_model <- "exp"
cov_pars <- c(4*10^8, 1.4)
nugget <- 4*10^8

# fitting using MML
mml_model <- likfit(geo_data, cov.model = cov_model, ini.cov.pars = cov_pars,
       fix.nugget = FALSE, nugget = nugget)
```

```
## kappa not used for the exponential correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
lines(mml_model, lty = 1, col="orange")

# fitting using REML
reml_model <- likfit(geo_data, cov.model = cov_model, ini.cov.pars = cov_pars,
       fix.nugget = FALSE, nugget = nugget, lik.method = "RML")
```

```
## kappa not used for the exponential correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
lines(reml_model, lty = 1, col = "purple")

# legend
legend(
  "bottomright",
  legend = c("MML", "REML"),
  col = c("orange", "purple"),
  lty = 1
)
```




```
# fitting a model to the empirical semivariogram (robust) - gaussian
plot(empirical_semivariogram_r)

cov_model <- "gaussian"
cov_pars <- c(4*10^8, 1.4)
nugget <- 4*10^8

# fitting by eye
lines.variomodel(cov.model = cov_model, cov.pars = cov_pars, nugget = nugget)

# fitting using equal weights
equal_weights_model <- variofit(empirical_semivariogram_r, cov.model = cov_model, ini.cov.pars = cov_pa
```

Fitting a Gaussian Model

```
## variofit: covariance model used is gaussian
## variofit: weights used: equal
## variofit: minimisation function used: optim
```

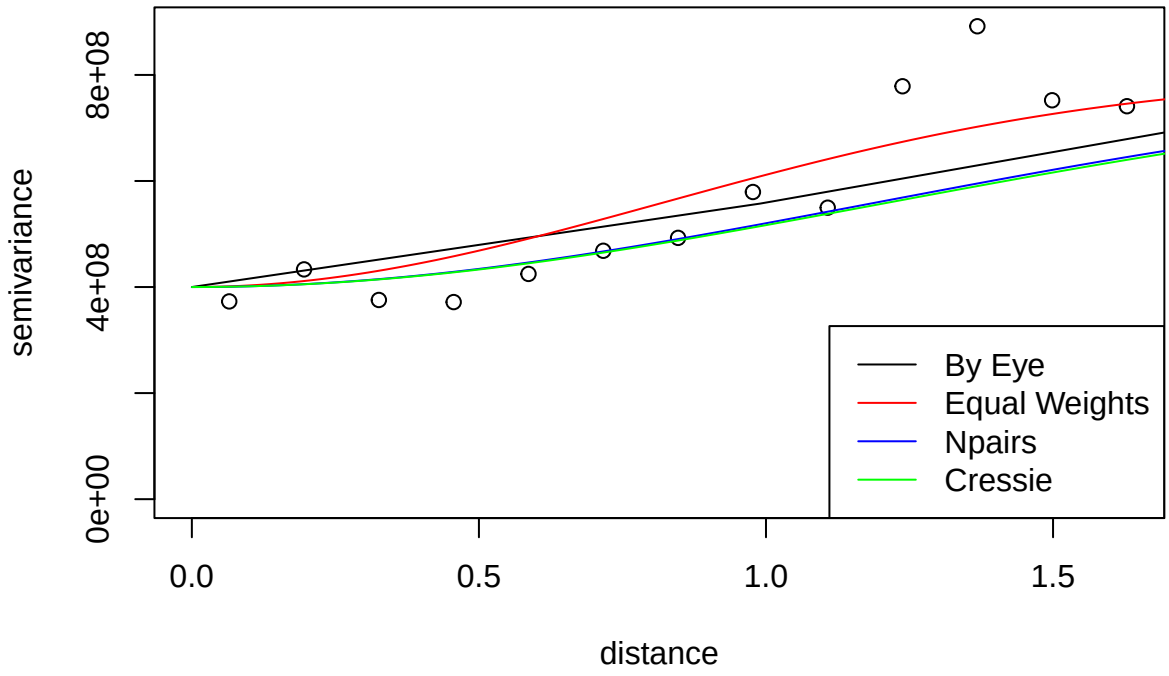
```
lines(equal_weights_model, lty = 1, col = "red")

# fitting using npairs weights
npairs_weights_model <- variofit(empirical_semivariogram_r, cov.model = cov_model, ini.cov.pars = cov_p
```

```
## variofit: covariance model used is gaussian
## variofit: weights used: npairs
## variofit: minimisation function used: optim
```

```
lines(npairs_weights_model, lty = 1, col = "blue")

# fitting using cressie's weights
cressies_weights_model <- variofit(empirical_semivariogram_r, cov.model = cov_model, ini.cov.pars = cov
```

```
## variofit: covariance model used is gaussian
## variofit: weights used: cressie
## variofit: minimisation function used: optim
```

```
lines(cressies_weights_model, lty = 1, col = "green")

# legend
legend(
  "bottomright",
  legend = c("By Eye", "Equal Weights", "Npairs", "Cressie"),
  col = c("black", "red", "blue", "green"),
  lty = 1
)
```



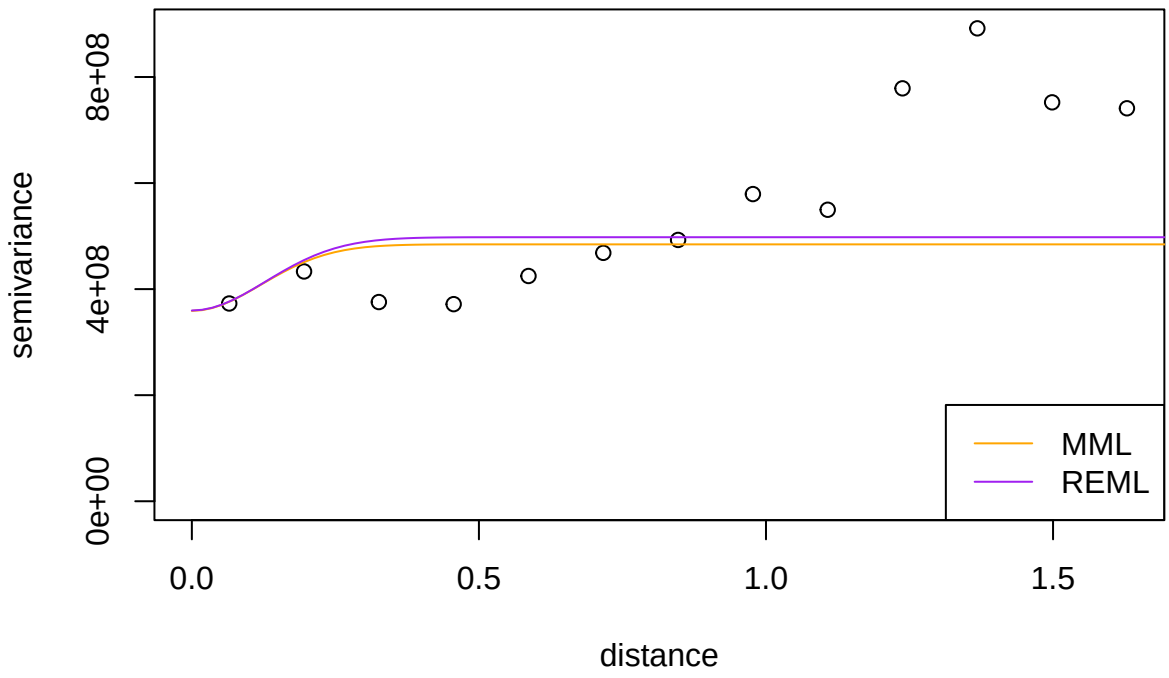
```r
# fitting a model using MLE - gaussian
# although the empirical semivariogram is not used for fitting, we would like to observe it's fit to th

plot(empirical_semivariogram_r)

cov_model <- "gauss"
cov_pars <- c(4*10^8, 1.4)
nugget <- 4*10^8

# fitting using MML
mml_model <- likfit(geo_data, cov.model = cov_model, ini.cov.pars = cov_pars,
       fix.nugget = FALSE, nugget = nugget)
```

```
## kappa not used for the gaussian correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
lines(mml_model, lty = 1, col="orange")

# fitting using REML
reml_model <- likfit(geo_data, cov.model = cov_model, ini.cov.pars = cov_pars,
       fix.nugget = FALSE, nugget = nugget, lik.method = "RML")
```

```
## kappa not used for the gaussian correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

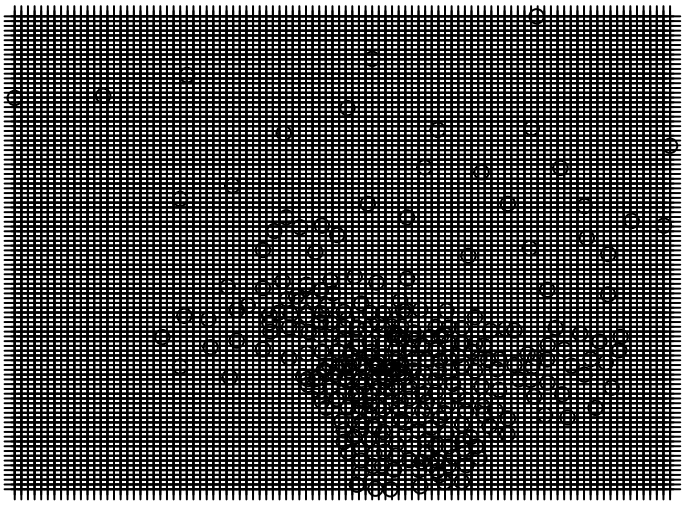
```r
lines(reml_model, lty = 1, col = "purple")

# legend
legend(
  "bottomright",
  legend = c("MML", "REML"),
  col = c("orange", "purple"),
  lty = 1
)
```



Spatial Prediction

Predictions on a dense grid using the inverse distance interpolation method

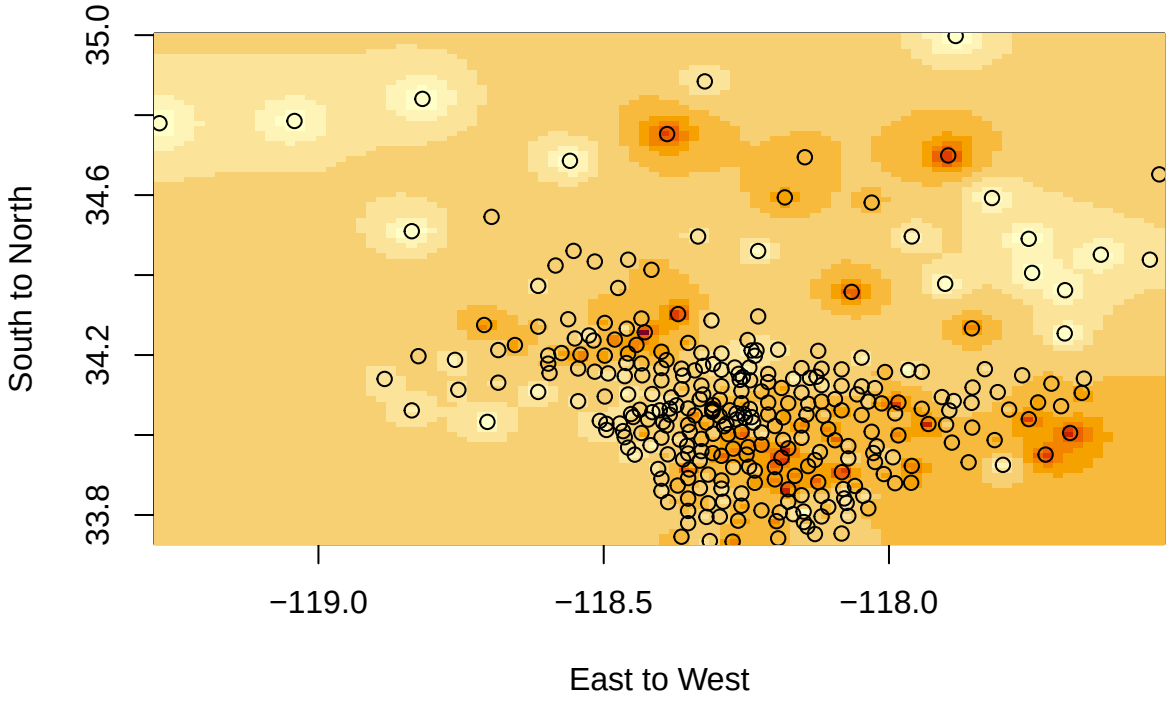
```
# create grid
x_min <- min(target_data$longitude)
x_max <- max(target_data$longitude)
y_min <- min(target_data$latitude)
y_max <- max(target_data$latitude)

x_values <- seq(x_min, x_max, length.out = 100)
y_values <- seq(y_min, y_max, length.out = 100)

xy_grid <- expand.grid(longitude = x_values, latitude = y_values)

coordinates(xy_grid) <- ~ longitude + latitude

# plot grid
plot(xy_grid)

# plot locations of observed data on the grid
points(target_data)
```



```r
# predict data points using grid as input
idw_pred <- idw(formula = population ~ 1, locations = ~ longitude + latitude, target_data, xy_grid)$var
```

```
## [inverse distance weighted interpolation]
```

```r
# convert vector to matrix
idw_pred_matrix <- matrix(idw_pred, length(x_values), length(y_values))

# plot predicted values
image(x_values, y_values, idw_pred_matrix, xlab = "East to West", ylab = "South to North")

# plot locations of observed data on the grid
points(target_data)
```



Predictions using different types of kriging

```
# variogram model to use for kriging
model <- equal_weights_model
```

```
# simple kriging
simple_krig <- krige.conv(geo_data, locations = as.data.frame(xy_grid), krige = krige.control(type.krig
```

Simple Kriging

```
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
```

```
head(simple_krig$predict)
```

```
## [1] 17160.60 17236.37 17326.57 17431.29 17550.59 17684.53
```

```
# ordinary kriging
ordinary_krig <- krige.conv(geo_data, locations = as.data.frame(xy_grid), krige = krige.control(type.kr
```

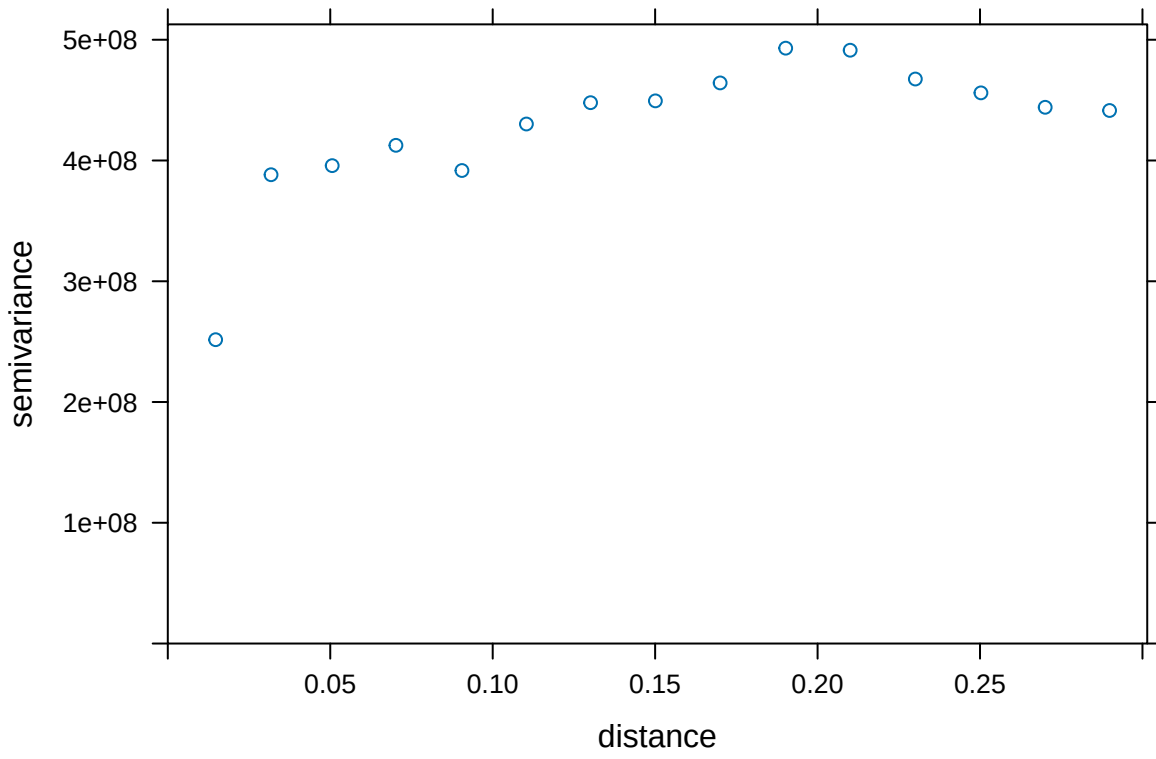
Ordinary Kriging

```
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
```

```
head(ordinary_krig$predict)
```

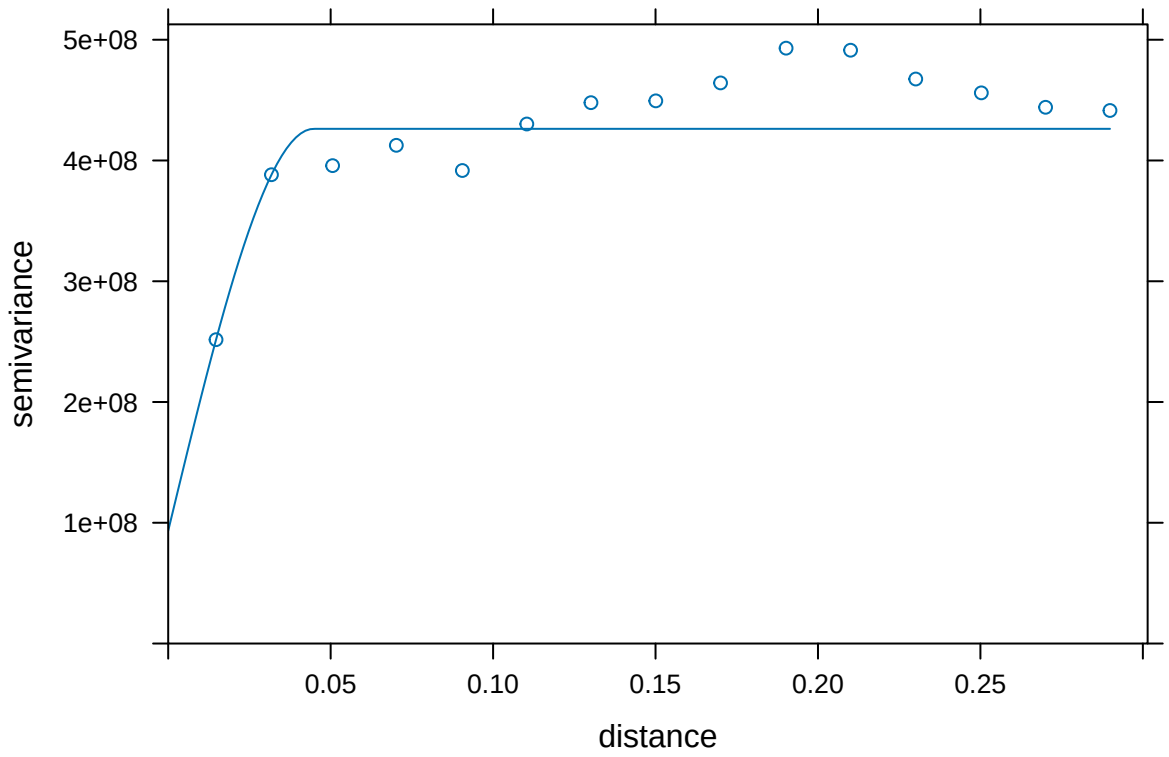
```
## [1] 12014.17 12216.72 12432.64 12661.95 12904.67 13160.78
```

```
# fit a variogram model with the assumption of trend

# convert to gstat object
g_stat <- gstat(formula = population ~ longitude + latitude, locations = ~ longitude + latitude, data =

# compute empirical semivariogram
empirical_semivariogram_trend <- variogram(g_stat, cutoff = 0.3)
plot(empirical_semivariogram_trend)
```



Universal Kriging

```r
# fit a variogram model
trend_model <- fit.variogram(empirical_semivariogram_trend, vgm(5*10^8, "Sph", 0.2, 2*10^8))
plot(empirical_semivariogram_trend, trend_model)
```




```r
# universal kriging
univ_krig <- krige(formula = population ~ longitude + latitude, locations = ~ longitude + latitude, mod
```

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

```r
head(univ_krig$var1.pred)
```

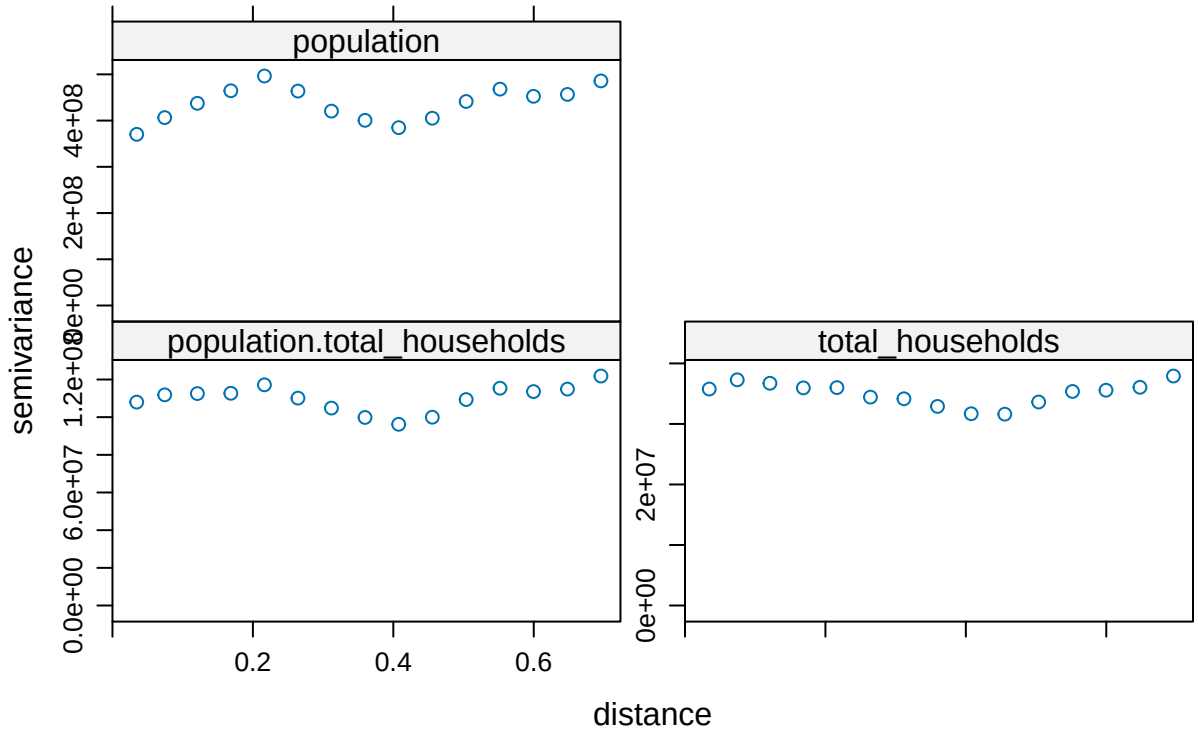
```
## [1] 34200.77 34331.40 34462.03 34592.66 34723.29 34853.92
```

```r
# compute correlation of population with each of the other variables
variable_data <- data[, variables]
correlations <- cor(variable_data, use = "pairwise.complete.obs")["population", ]
correlations
```

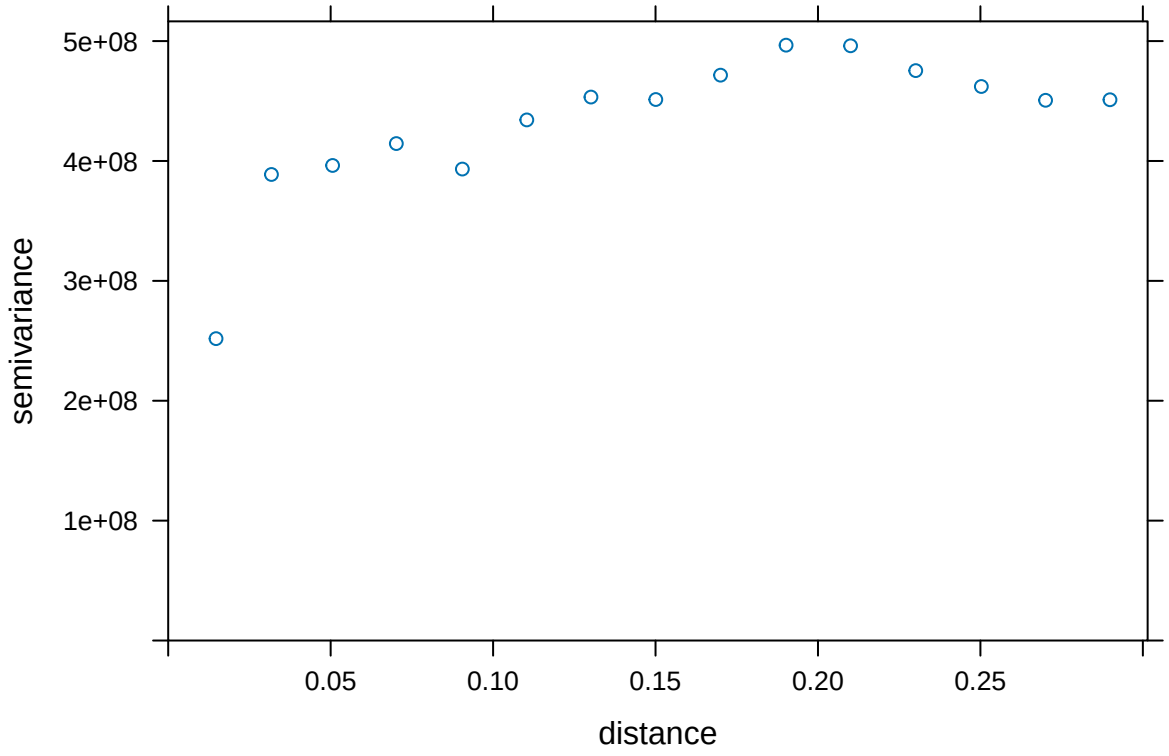
Co-kriging

```
##          population         median_age       percent_male     percent_female
##           1.0000000         -0.4712809         -0.1429971          0.1429971
##   total_households avg_household_size
##           0.9022298          0.5326180
```

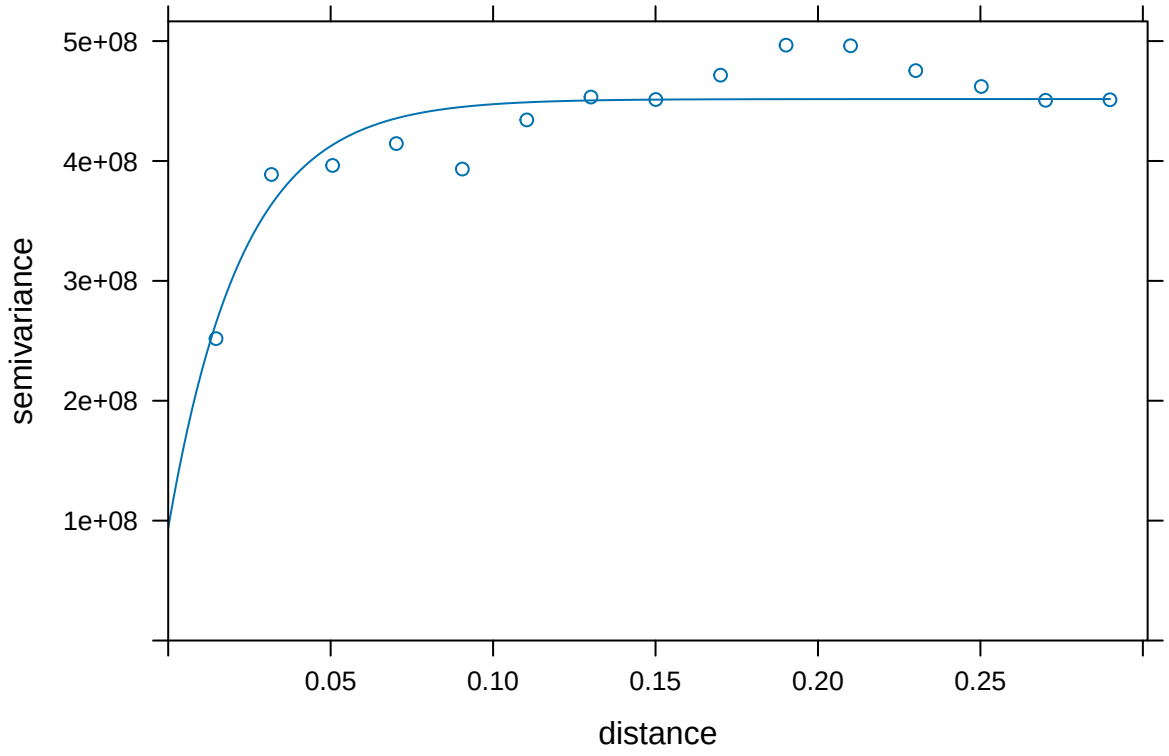
```r
# conduct co-kriging with population as the target variable and two other co-located variables: total_h

# create gstat object including all three variables
g_stat <- gstat(id="population", formula = population ~ 1, locations = ~ longitude + latitude, data = da
g_stat <- gstat(g_stat, id = "total_households", formula = total_households ~ longitude + latitude, loca

# compute and plot the semivariograms for each of the variables
empirical_cross_semivariogram <- variogram(g_stat)
plot(empirical_cross_semivariogram)
```



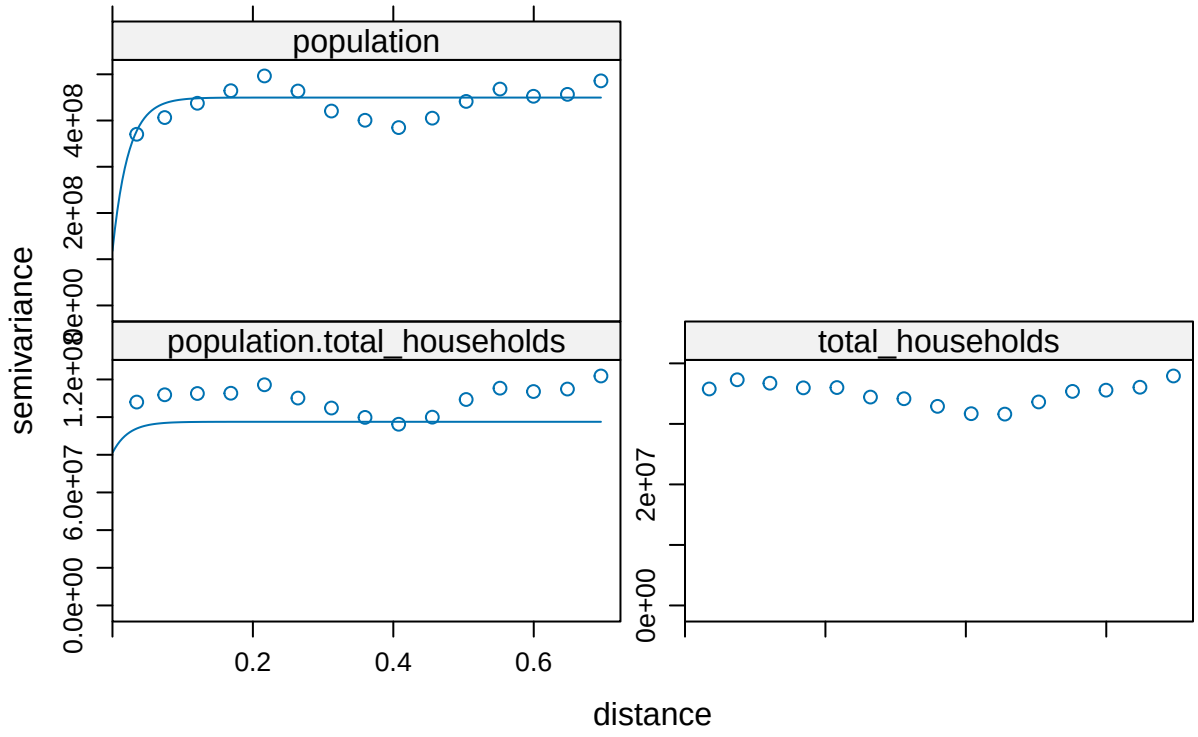

```r
# fit a variogram model to the sample variogram of the target variable
g_stat_target <- gstat(id = "population", formula = population ~ 1, locations = ~ longitude + latitude,
empirical_semivariogram_target <- variogram(g_stat_target, cutoff = 0.3)
plot(empirical_semivariogram_target)
```




```
model_target <- fit.variogram(empirical_semivariogram_target, vgm(4*10^8, "Exp", 0.2, 0), fit.method = 1
plot(empirical_semivariogram_target, model_target)
```




```
# fit a variogram model to the sample cross-variogram
model_cross <- fit.lmc(empirical_cross_semivariogram, g_stat, model = model_target)
plot(empirical_cross_semivariogram, model_cross)
```




```
# co-kriging
co_krig <- predict(model_cross, xy_grid)
```

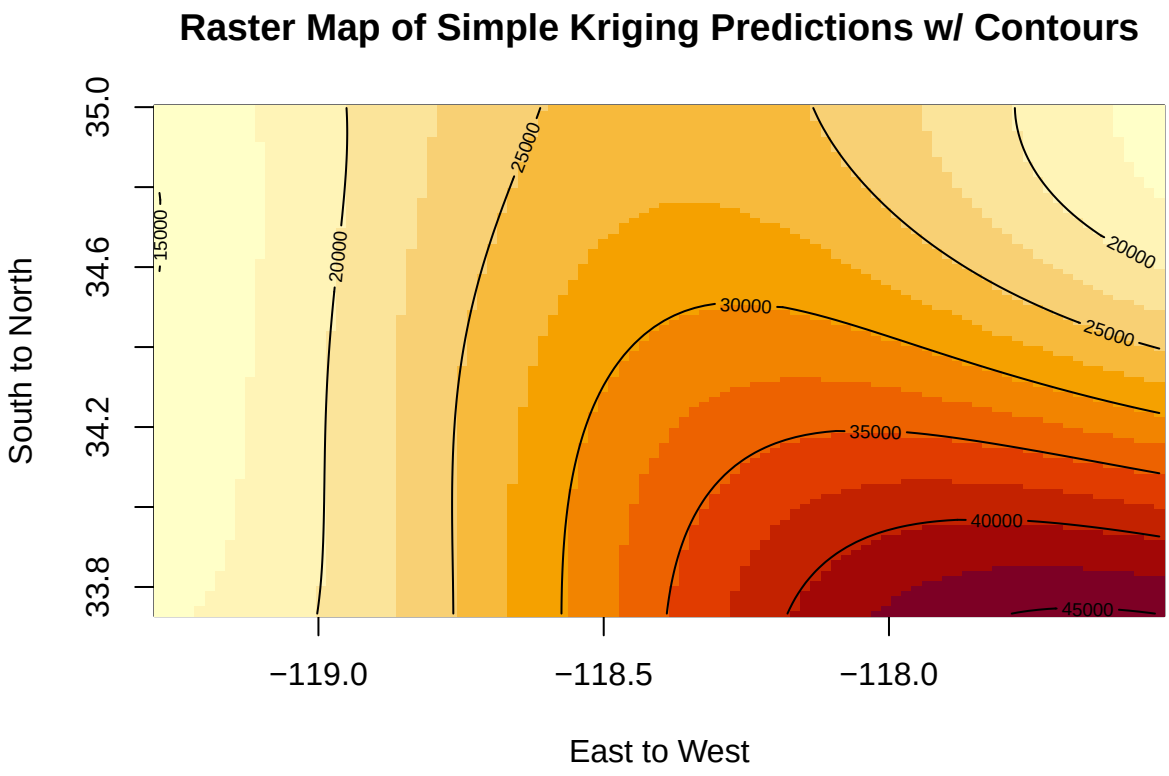
```
## Linear Model of Coregionalization found. Good.
## [using universal cokriging]
```

```
head(co_krig$population.pred)
```

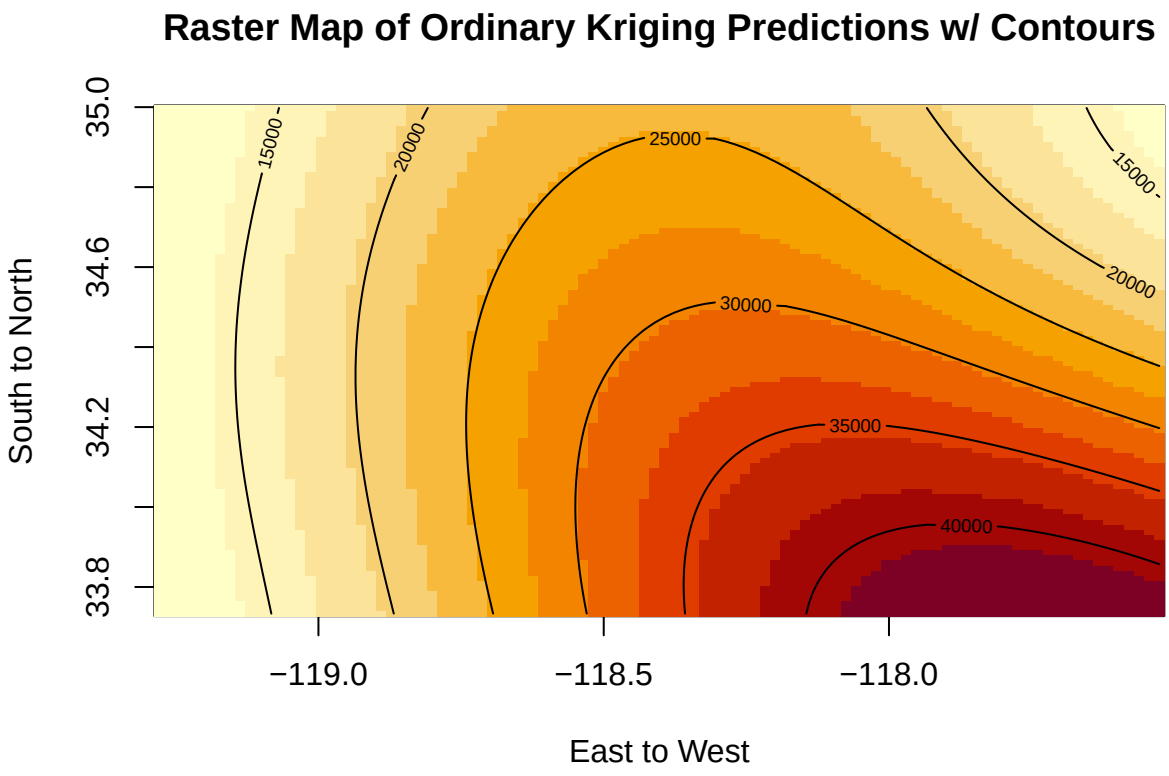
```
## [1] 33704.69 33704.69 33704.69 33704.69 33704.69 33704.69
```

Raster Maps and add Contours, for each set of kriging predictions

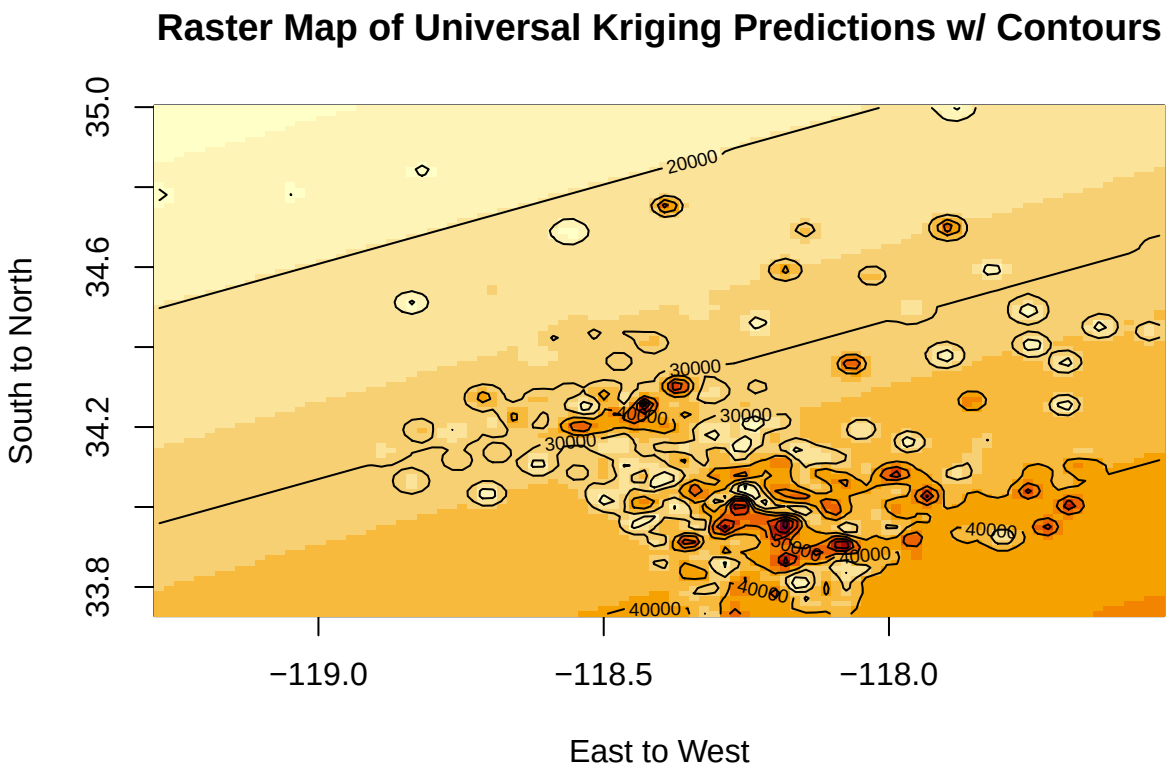
```
# simple kriging raster map
prediction_matrix <- matrix(simple_krig$predict, length(x_values), length(y_values))
image(x_values, y_values, prediction_matrix, xlab = "East to West", ylab = "South to North", main = pas
contour(x_values, y_values, prediction_matrix, color = "black", add = TRUE)
```



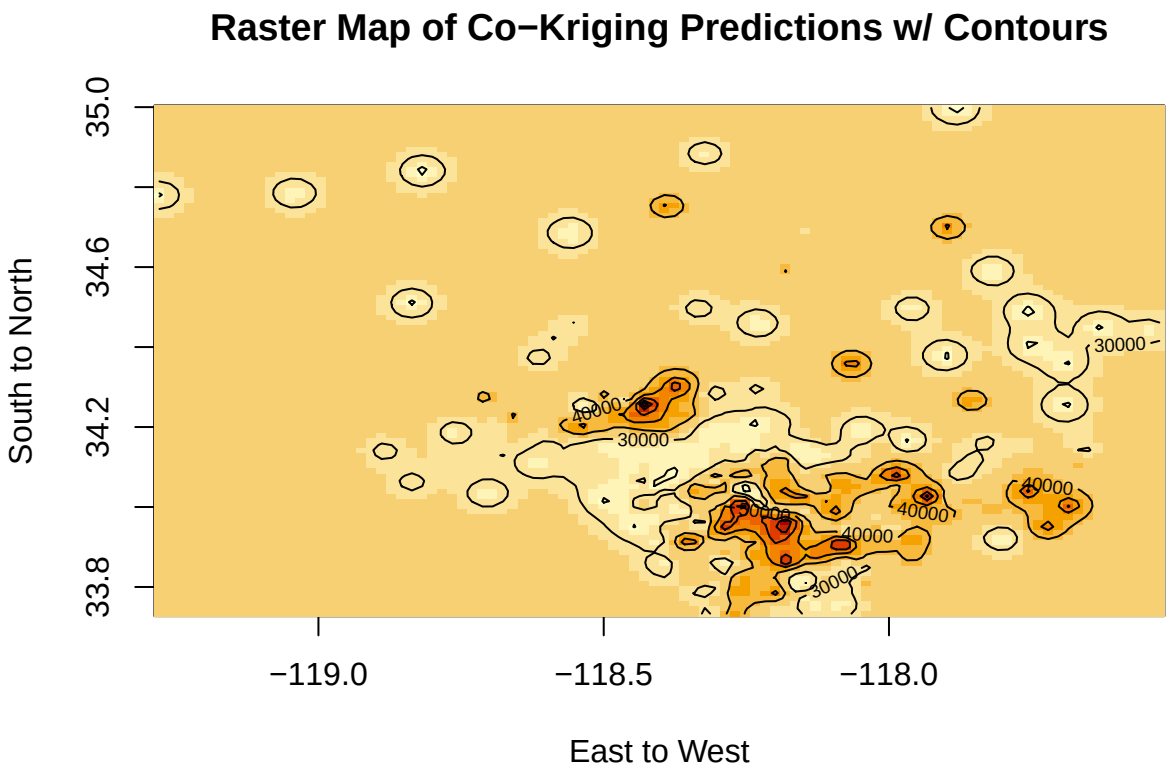

```r
# ordinary kriging raster map
prediction_matrix <- matrix(ordinary_krig$predict, length(x_values), length(y_values))
image(x_values, y_values, prediction_matrix, xlab = "East to West", ylab = "South to North", main = pas
contour(x_values, y_values, prediction_matrix, color = "black", add = TRUE)
```




```r
# universal kriging raster map
prediction_matrix <- matrix(univ_krig$var1.pred, length(x_values), length(y_values))
image(x_values, y_values, prediction_matrix, xlab = "East to West", ylab = "South to North", main = pas
contour(x_values, y_values, prediction_matrix, color = "black", add = TRUE)
```




```r
# co-kriging raster map
prediction_matrix <- matrix(co_krig$population.pred, length(x_values), length(y_values))
image(x_values, y_values, prediction_matrix, xlab = "East to West", ylab = "South to North", main = pas
contour(x_values, y_values, prediction_matrix, color = "black", add = TRUE)
```


Raster Map of Co−Kriging Predictions w/ Contours
South to North
35.0
34.6
34.2
33.8
−119.0
−118.5
−118.0
East to West
30000
40000
30000
40000
40000
40000
30000
30000